МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА АЭРОКОСМИЧЕСКИХ КОМПЬЮТЕРНЫХ И ПРОГРАММНЫХ СИСТЕМ

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доцент, к.т.н., доцент		С.В. Горбачев
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

«ПРОЕКТИРОВАНИЕ ЛВС С ЛОКАЛЬНО-ПРИОРИТЕТНЫМ ДОСТУПОМ»

по дисциплине: «СЕТИ И ТЕЛЕКОМУНИКАЦИИ»

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ_ГР. № 1842		Я.А.Вишневский
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022 г.

Министерство науки и высшего образования РФ
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Факультет № ____1_____ Кафедра N __14____

ЗАДАНИЕ

по курсовому проектированию по дисциплине «Сети ЭВМ и телекоммуникации»
на тему: «Проектирование сетевого контроллера кольцевой ЛВС»_____
Выдано студенту Вишневскому Я. А. группа № 1842
__8___февраля__2021 г. Срок исполнения _20 апреля 2022 г.

1. ТЕХНИЧЕСКИЕ УСЛОВИЯ

Вариант 4.

1) Поток симметричен._____

2) Тактовая частота сдвига Fт=2 МГц._____

3) Разрядность буферного регистра последовательного интерфейса ω=8._____

4) Расстояния между станциями одинаковы._____

5) Направление передачи – в сторону увеличения номеров._____

6) Количество станций N = 100._____

7) Интенсивность входного потока λ_t = 20 1/с._____

8) Длина кадра b = 800 бит._____

9) Приоритет станции._____

10) Длина сети L = 1 км._____

11) Метод КО (программная модель) - CRC16._____

12) Алгоритм источника единичной передачи типа «точка-точка» (источник ЕдП).___

13) Определение вероятности прохождения кадра от станции №90 до станции №60.___

14) Определение задержки между станциями №70 и №10._____

2. СОДЕРЖАНИЕ ПРОЕКТА

– Принципы работы ЛВС с детерминированным доступом;

– краткое обоснование модели кольца со вставными регистрами;

– определение функциональных зависимостей основных характеристик проектируемой ЛВС: вероятность Q(i,k), средняя величина задержки передачи кадра Tf(i,j;

– описание используемого формата кадра и метода контроля ошибок;

– разработка структуры сетевого контроллера и описание графа его состояний;

– разработка протокола передачи кадра подуровня MAC и описание заданного алгоритма работы узла сети;

– программная модель заданного метода контроля ошибок для сетевого контроллера;

– анализ результатов.

– 3. ОФОРМЛЕНИЕ ПРОЕКТА

1. Пояснительная записка с рисунками и расчетными таблицами
2. Чертежи
   - блок-схемы ЛВС для заданных вариантов расчетов параметров;
   - структурная схема сетевого микроконтроллера;
   - граф-схема состояний сетевого микроконтроллера;
   - граф-схема алгоритма процедуры передачи кадра в сети;
3. Приложение
   - листинг программы моделирования, скриншоты результатов моделирования.

4. УКАЗАНИЯ

Пояснительная записка должна быть выполнена на листах формата 210х297 чернилами или напечатана на принтере; используемые расчетные формулы приводить в буквенном обозначении с кратким объяснением, а затем с числовыми значениями.

Выполненные проекты не возвращаются.

ЛИТЕРАТУРА

1. С.В. Горбачев, Ю.Д . Крылов. ПРОЕКТИРОВАНИЕ УПРАВЛЯЮЩИХ ЛОКАЛЬНЫХ СЕТЕЙ НА ОСНОВЕ МИКРОКОНТРОЛЛЕРОВ / Метод. указания для самостоятельной работы по курсовому проектированию. ГУАП, 2022, в эл. виде.
2. Компьютерные сети. Принципы, технологии, протоколы. Учебник / В.Г. Олифер, Н.А. Олифер. – СПб: Издательство «Питер», 2016. – 672 с.
3. Ю. Блэк. Сети ЭВМ: протоколы, стандарты, интерфейсы. - М.: Мир, 1990.
4. В.К. Щербо и др. Справочник. Стандарты по локальным вычислительным сетям. - М.: Радио и связь, 1990.
5. Д. Дэвис, Д. Барбер, У. Прайс, С. Соломонидес. Вычислительные сети и сетевые протоколы. - М.: Мир, 1982.

Руководитель проекта _____ /С.В. Горбачев/
(подпись)

Задание принял к исполнению _____ /Я.А. Вишневский/
(подпись)

Оглавление

Описание принципов работы ЛВС с детерминированным доступом и обоснование модели кольца со вставными регистрами

Описание принципов работы ЛВС с детерминированным доступом

При детерминированных методах доступа в сети с топологией «кольцо» все компьютеры осуществляют взаимодействие с моноканалом через свои кольцевые адаптеры (станции). Детерминированный доступ основан на поочередном предоставлении права на передачу кадра.

Существует несколько подходов к реализации детерминированного доступа к моноканалу:

- пропорциональный доступ;
- приоритетный доступ;
- локально-приоритетный доступ.

Пропорциональный доступ исключает возможность конфликта источников за счет задания очередности доступа абонентов к каналу. Переключение очередности производится либо с помощью признаков разрешения доступа (маркеров), либо с помощью синхросигналов.

При одном из способов приоритетного доступа источники сначала передают в канал управляющую информацию, задающую их приоритеты. Источники используют процедуру децентрализованного кодового управления для выявления источника с наибольшим приоритетом, который передает данные уже при отсутствии конфликта. Более известным и широко используемым способом приоритетного доступа является маркерный доступ с приоритетами, реализованный фирмой IBM в ЛВС Token Ring и затем признанный стандартом IEEE 802.5.

Сети Token Ring используют систему приоритетов, которая позволяет некоторым абонентам с высоким приоритетом, более часто пользоваться собой сетью. В Token Ring маркер, представляющий собой короткий служебный кадр, содержит два поля, которые управляют приоритетом: поле приоритетов и поле резервирования. Только абоненты, приоритет которых равен или выше приоритета, содержащегося в маркере, могут завладеть им. После того, как маркер захвачен и изменен (в результате чего он превратился в информационный кадр), только станции, приоритет которых равен или выше приоритета передающей станции, могут зарезервировать маркер для следующего цикла передачи. При восстановлении маркера в него включается более высокий приоритет данной резервирующей станции. Станции, которые повышают уровень приоритета маркера, должны восстановить предыдущий уровень приоритета после завершения передачи своего кадра.

При локально-приоритетном доступе любой источник сравнивает приоритет своего кадра с приоритетом кадра, передающегося по каналу в месте его подключения. Дальше по каналу источник передает кадр с более высоким приоритетом, а другой оставляет у себя для последующей передачи. При отсутствии передачи в канале источник передает свой кадр независимо от его приоритета. Обычно реализуются крайние возможности подхода: любой кадр источника имеет либо более высокий, либо более низкий приоритет, чем любой кадр в кольце.

Основные способы локально–приоритетного доступа – сегментированные кольца, кольца типа Cambridge Ring (CR), с переменной задержкой и со вставкой регистров.

Сегментированные кольца и кольца с переменной задержкой могут иметь одну или более кольцевых магистралей.

При реализации методов локально-приоритетного доступа кольцевой адаптер каждого источника должен прослушивать моноканал в месте его подключения и при отсутствии в нем передачи данных может начать передачу своего кадра. Кроме того, источник может заменять приходящий по кольцу кадр на свой кадр, если его приоритет выше (в кольцах с переменной задержкой и вставкой регистров). В кратных сегментированных кольцах приемник имеет возможность адресованные ему кадры изымать из кольца.

Обоснование модели кольца со вставными регистрами

В кольцах со вставными регистрами (буферами) конфликты между пакетами (кадрами), находящимися у станций и готовыми к передаче, и пакетами, идущими по кольцу, динамически разрешаются путем включения последовательных (сдвиговых) регистров в кольцо. Все регистры (буферы) включены внутрь адаптеров кольца. На рис.2 показана структура адаптера со вставным буфером транзита.

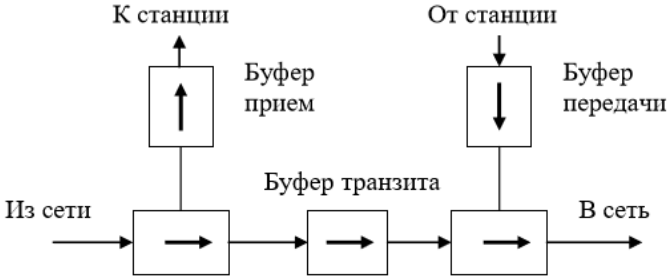

Рисунок 1 – Структура кольцевого адаптера со вставным буфером транзита

Пусть станция имеет канал данных, который временно хранится в ее буфере передачи. Станция начинает передачу в момент времени, зависящий от принятого способа функционирования, т.е. от принятой системы приоритетов при передаче данных, при этом различают приоритет кольца и приоритет станции.

Если же станция не передает свои собственные пакеты, содержимое вставного буфера транзита передается следующей станции.

На рис.3 изображена модель кольца с буферами транзита при наличии трех станций. Для описания функционирования системы положим, что пакет, сформированный станцией 1 и предназначенный для станции 3, проходит через представленную модель.

После своей генерации пакет помещается в буфер передачи станции 1, где он должен ждать своей очереди на передачу. Пакеты, идущие по кольцу, запоминаются в буфере транзита и вставляются в расписание в соответствии с принятой дисциплиной обслуживания: приоритет кольца или приоритет станции. Начало передачи пакета означает, что началось обслуживание и пакет поступает в блок задержки станции 2. Это является важнейшим свойством данной модели.

Блоки задержки $\tau 1$, $\tau 2$ и $\tau 3$ учитывают задержку передачи, необходимую для дешифрации адреса, и время распространения сигнала по соответствующей дуге кольца.

В блоке задержки станции 2 пакет считается задержанным на постоянное время $\tau 2$, а затем он помещается в буфер транзита станции 2. Считается, что пакет поступает в блок задержки станций J, когда он передается по дуге 2-3.

Проследовав постоянную задержку $\tau 3$, пакет поступает к станции 3, т.е. в буфер приема, не обозначенный на рис.3 При этом пакет покидает модель.

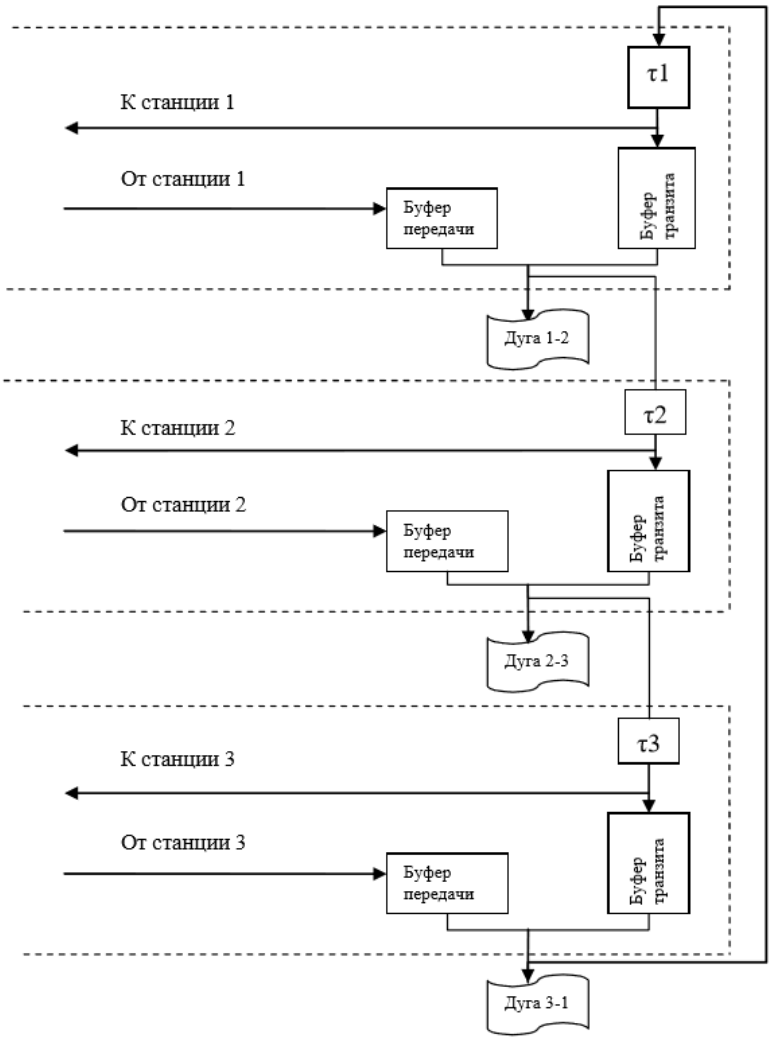

Рисунок 2 – Модель кольца с буферами транзита при наличии трех станций

Достоинства модели кольца со вставными регистрами:

− Никакая группа источников не может монополизировать кольцо, поэтому любому источнику обеспечивается гарантированное время доставки пакета к приемнику.

− В отличие от сетей с опросом, нет простаивания абонентов, содержащих готовые для передачи данные.

− Учитывается приоритет (приоритет станции или кольца).

Анализ эквивалентной модели станции ЛВС

Для простоты анализа принимаем, что емкость приемного буфера достаточно высока, чтобы не влиять на протекание потока пакетов.

Обозначим через Tf(i,j) задержку передачи пакета как временного интервала от момента генерации пакета на станции i до его получения на станции j.

Будем использовать следующие допущения:

1. Входные потоки пакетов от всех станций являются пуассоновскими с интенсивностями λt(1),…,λt(s),…,λt(N), где s – номер очередной станции, N – общее число станций.

2. Время передачи пакета в кольцо Tp может быть распределено по произвольному закону, а распределение длин пакетов, сформированных различными станциями, идентичны.

3. Отношения между источниками и получателями пакетов определяются произвольной матрицей выбора маршрута |P(i,j)|, где P(i,j) – вероятность того, что пакет, сформированный на станции i, предназначен для станции j.

4. Задержка ti, кольцевого адаптера i постоянна. Для удобства в величину τi также включается задержка распространения сигналов, передаваемых между адаптерами (i-1) и i.

Проведем анализ представленной модели.

Он основан на декомпозиции полной модели кольца в подходящую подмодель.

Затем производится анализ подмодели, и вычисляются значения задержек в полной модели на основании результатов, полученных на подмоделях.

Простейший путь декомпозиции полной модели – это рассмотрение подмодели, содержащей одну дугу кольца, буфер передачи, буфер транзита и блок задержки.

Анализ подмодели упрощается при введении следующих предположений:

- входной поток к буферу транзита является пуассоновским;
- время поступления пакета и время обработки (т.е. передачи) взаимно независимы.

Действительное время задержки пакета данных дается суммой трех слагаемых: времени задержек в очереди в буфере передачи; времени передачи, времени суммарных задержек в адаптерах кольца.

Необходимо отметить два обстоятельства:

- очередь имеется только по посылающей станции;
- подмодель имеет ненулевую задержку в каждом адаптере кольца,

следовательно, сумма всех задержек в очереди зависит от среднего числа адаптеров, которые должен пройти пакет.

Кроме предположений, введенных ранее, дополнительно примем следующее:

- момент поступления пакета на блок задержки станций идентичен началу его передачи по дуге (i-1)-i;
- время распространения по различным дугам одинаково.

Это достигается включением в подмодель станции i (рис.4) следующих компонентов: очередей на станции (i-1); дуги i-(i+1); очередей на станции (i-1); дуги (i-1)-i.

Подмодель станции i используется только для определения величин задержек очередей в буфере передачи Wt(i) и в буфере транзита Wv (i), но не для определения задержек очередей на станции (i-1). Целью включения очередей станции (i-1) и дуги (i-1)-1 в подмодель станции i является точное описание действительного процесса поступления пакетов на станцию i и его взаимозависимости с процессом передачи пакетов по дуге i-(i+1).

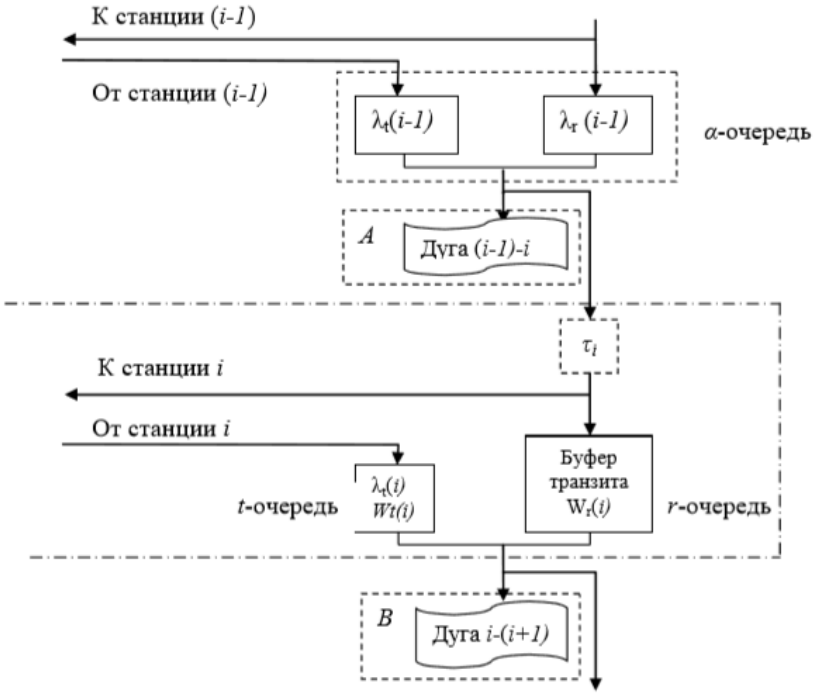

Рисунок 3 – Подмодель станции с очередями

Интерфейс подмодели станции i к общей модели кольца дается определением входного потока к буферу транзита станции (i – 1). Предполагается, что этот поток пуассоновский.

Произведем анализ подмодели станции i.

Для того чтобы определить задержки в подмодели станции i, преобразуем эту подмодель в упрощенную подмодель станции i. Это делается с помощью трех шагов:

- без потери общности полагаем, что задержки τi равны нулю, поскольку задержка приводит просто к временному сдвигу;
- все пакеты, поступающие на станцию i, рассматриваются как транзитные пакеты;
- входной процесс поступления пакетов к входному (фиктивному) буферу станции i не подчиняется действию обычных правил приоритета, применяемых к буферу передачи и буферу транзита станции (i-1).

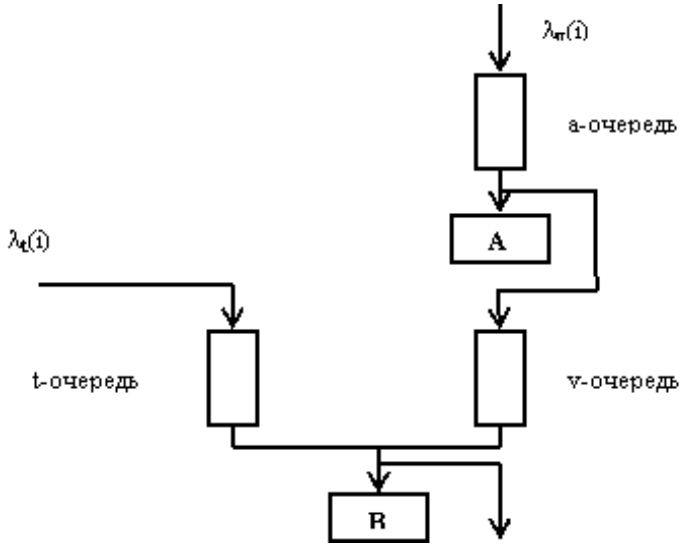

Рисунок 4 – Упрощенная подмодель станции i

Мы можем, следовательно, соединить оба буфера на станции (i-1) в один буфер и организовать одну очередь. Она называется a-очередью в упрощенной подмодели (рис.5). Объединенный входной поток пакетов в очереди α будет пуассоновским с интенсивностью $\lambda_v(i)$ Все другие компоненты подмодели станции i являются неизменными в упрощенной подмодели станции i (рис.5). При этом дуги кольца являются обслуживающими приборами:

- обслуживающий прибор A описывает действия дуги (i-l)-i;
- обслуживающий прибор B – дуги i – (i+1).

Для простоты описания буфер передачи станции i назван "t-очередью", а буфер транзита – " V-очередью".

Основные свойства упрощенной подмодели станции i следующие:

- пакеты, полученные для приема в V-очередь, начинают также обслуживаться прибором A;
- время обслуживания пакетов в обслуживающих приборах A и B идентично.

Входной поток, поступающий в V-очередь на рис.5 является комплексным, так как его составные части поступают от буфера передачи и буфера транзита станции (i-1). Суммарная интенсивность этого потока $\lambda_v(i)$.

Необходимо отметить, что наличие прибора обслуживания в подмодели не оказывает влияния на процесс обслуживания в приборе B. Таким образом, обслуживающий прибор B будет вести себя так же, как если бы пакеты кольца вместо того чтобы поступать в a-очередь, направляются прямо к V-очереди, как это показано в «эквивалентной модели» (рис.6).

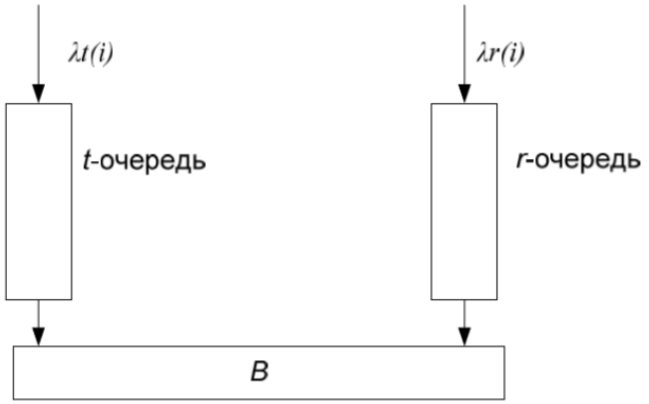

Рисунок 5 – «Эквивалентная модель»

В упрощенной подмодели (рис.6) обслуживание пакета в приборе A начинается либо немного раньше, чем в обслуживании в приборе B (при наличии пакетов в очереди V), либо точно в тот же самый момент времени (если нет пакетов в очереди V).

Следовательно, если допускается, что модели (рис.5 и рис.6) управляются идентичными входными потоками, обслуживающий прибор B (рис.5) всегда будет обрабатывать те же самые пакеты из v - или t -очередей, как и обслуживающий прибор на рис.6, если рассматривается начало нового обслуживания, т. е. если нет остаточных пакетов в V-очереди (рис.5).

Дисциплина обслуживания пакетов в каждой очереди – "первым пришел – первым обслужен" (FIFO). Поэтому процессы обработки в обслуживающем приборе B в подмоделях (рис.5 и 6) идентичны. Таким образом, задержка $d_v(i)$ от момента поступления пакета в a-очередь (рис.5) до того, как его обслуживание начнется в обслуживающем приборе B, равна задержке в объединенной V-очереди (рис.6). Итак, «эквивалентная подмодель» станции i – это система массового обслуживания класса

M|G|l (пуассоновский входной поток, произвольное время обслуживания, один обслуживающий прибор), свойства которой хорошо известны.

Определение функциональных зависимостей основных характеристик проектируемой ЛВС

Исходные данные:

- Поток симметричен;
- Тактовая частота сдвига F_T = 2 МГц;
- Разрядность буферного регистра последовательного интерфейса ω=8;
- Расстояния между станциями одинаковы;
- Направление передачи – в сторону увеличения номеров;
- N = 100 – число станций ЛВС;
- λ_t = 20 (1/с) – интенсивность входного потока кадров;
- b = 800 (бит) – длина кадра;
- Приоритет – s (приоритет станции);
- L = 1 (км) – длина сети;
- Метод КО – CRC16 (циклический контроль с избыточностью ошибок в кадре - 16-разр. полином (Ethernet);
- Алгоритм контроля ошибок и программная модель – алгоритм источника единичной передачи типа «точка-точка» (ЕдП)

Найти:

- Среднюю величину задержки передачи кадра T_f (70, 10); i = 70 j=10;
- Интенсивность поступления кадров в буфер транзита станции ЛВС в зависимости от вероятности Q (90, 60); i = 90 k=60;

Теоретические материалы:

1) Для определения интенсивности потока пакетов в конце у каждой станции $\lambda_v(i)$ необходимо определить вероятность Q(i,k) того, что пакет, сформированный на станции i, проходит транзитом через станцию k. При этом:

$$Q(i,k) = \sum_{j \in I(i,k)} P(i,j) \qquad (1)$$

для всех j, которые определены на множестве $I(i,k)$. Здесь:

$$I(i,k) = \begin{cases} \{M - \{i < x \le k\}, & \text{если } i < k \\ X | k < x \le i, & \text{если } i > k \end{cases} \qquad (2)$$

где M – множество целых чисел $1 \le M \le N$; X – множество целых чисел $i < x \le k$; или X – множество целых чисел, $k < x \le i$.

Интенсивность поступления пакетов в буфер транзита может быть определена по формуле:

$$\lambda_r(k) = \sum_{i=1}^{N} \lambda_t(i) * Q(i,k) \qquad (3)$$

В случае симметричного потока:

$$\lambda_t(i) = \lambda_t = const \qquad (4)$$

$$\lambda_r = \lambda_t \sum_{i=1}^{N} Q(i,k) = \lambda_t * \frac{1}{N-1} * \left[\frac{N*N-N}{2} - (N-1)\right] \qquad (5)$$

2) Средняя величина задержки передачи пакета, сформированного на станции i и предназначенного станции j, определяется следующим выражением:

$$T_f(i,j) = W_t(i) + \sum_{n \in M(i,j)} (W_r(n) + \tau_n) + \tau_j + E[T_p] \qquad (6)$$

где $W_t(i)$ – среднее время ожидания в t-очереди у передающей станции, т.е. в буфере передачи.

$\sum(W_{r(n)} + \tau_n$ – суммарная задержка передачи пакета данных в буферах транзита и блоках задержки транзитных станций.

M(i,j) – множество транзитных станций между станциями i и j.

τ_j – время задержки у приемной станции;

E[Tp] – среднее время передачи в кольцо пакета данных.

$$M(i,j) = \begin{cases} \{i < x < j\}, \text{если } i < j \\ NS - \{j \le x \le i\}, \text{если } i > j \end{cases} \quad (7)$$

где *NS* – множество станций, $NS = \{1,2,3,\dots,N\}$

Величина τ_n находится как:

$$\tau_n = \tau_a + t_{nx} = \frac{\omega}{F_T} + \frac{L/N}{c} \quad (8)$$

где τ_a – активная составляющая задержки, $\tau_{nк}$ – задержка передачи по каналу, ω – разрядность буферного регистра последовательного интерфейса, F_T – тактовая частота сдвига, *L* – длина сети, *N* – число станций ЛВС, *c=200000 км/с* – скорость распространения сигнала по кабелю.

Ожидание начала обслуживания в t-очереди в случае приоритета станции (s) будет следующим:

$$W_t{}^s(i) = \frac{[\rho_r(i)+\rho_t(i)]*E[T_p{}^2]}{2*[1-\rho_r(i)]*E[T_p]} \quad (9)$$

Следовательно, ожидание начала обслуживания (передачи кадра)для r-очереди в случае приоритета станции (s)будет следующим:

$$W_r{}^{(s)}(i) = d_r{}^{(s)}(i) - W_a(i) = \frac{\rho_t(i)*(1+\rho_r(i)*\{1-\rho_t(i)-\rho_r(i)\})*E[T_p{}^2]}{2*[1-\rho_t(i)-\rho_r(i)]/[1-\rho_t(i)]*[1-\rho_r(i)]*E[T_p]} \quad (10).$$

Где:

$$\rho_t(i) = \lambda_t(i) * E[T_p] \quad (11)$$

$$\rho_r(i) = \lambda_v(i) * E[T_p] \quad (12)$$

Здесь $\rho_t(i)$, $\rho_r(i)$ – загрузки обслуживающего прибора, создаваемые t- и r-очередями соответственно; *Tp* – время обслуживания (передачи) пакета данных.

Среднее время передачи в кольцо пакета данных определяется как:

$$E[T_p] = \frac{b}{fd} \quad (13)$$

где *b* – длина кадра [бит], *fd* – скорость передачи кадра [бит/с].

Расчеты:

1) Для симметричного потока в системе, содержащей N=100 станций, получаем:

$$P(i,j) = \frac{1}{N-1} = \frac{1}{100-1} = \frac{1}{99}$$

$$P(i,i) = 0$$

Тогда:

$$[P(i,j)] = \begin{matrix} 0 & \frac{1}{99} & \frac{1}{99} & \cdots & \frac{1}{99} \\ \frac{1}{99} & 0 & \frac{1}{99} & \cdots & \frac{1}{99} \\ \frac{1}{99} & \frac{1}{99} & 0 & \cdots & \frac{1}{99} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{1}{99} & \frac{1}{99} & \frac{1}{99} & \cdots & 0 \end{matrix}$$

Найдем величину I (90, 60) по формуле (2), где M = {1, 2, 3, 4, … 100} и X = {61, 62, 63, 64, … 90}, так как $i \geq k$, т.е. $90 \geq 60$, то

$I\,(90{,}60) = \{\mathrm{X} \mid 60 < x \leq 90\} = \{61, 62, 63, 64, \ldots 90\}$

$|I| = 30$.

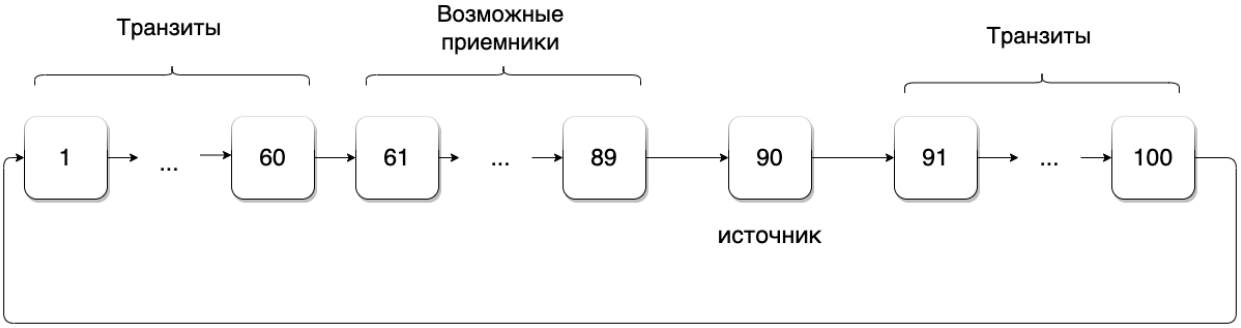

Рисунок 6 – Схема кольцевой ЛВС для задачи нахождения Q(90,60)

$$P(90;90) = 0;$$

$$P(90;61) = \cdots = P(90;89) = \frac{1}{99}$$

По формуле (1) находим Q (90,60):

Q (90, 60) = $P(90,61) + P(90,62) + \ldots + P(90,90) + = 29 * \frac{1}{99} + 0 = \frac{29}{99}$

В случае симметричного потока λ_t(i) = 20 1/с, тогда с учетом $P(i,i) = 0$, имеем λ_r, находящееся по формуле (5):

$$\lambda_r = \lambda_t * \frac{1}{N-1} * \left[\frac{N*N-N}{2} - (N-1)\right]$$

$$= 20 * \frac{1}{100-1} * \left[\frac{100*100-100}{2} - (100-1)\right] = 980\frac{1}{c}$$

2) По формуле (8) определяем τ_n:

$\tau_n = \tau_a + t_{nx} = \frac{\omega}{F_T} + \frac{L/N}{c} = \frac{8}{2 \cdot 10^6} + \frac{1/100}{2 \cdot 10^5} = 4{,}05 \cdot 10^{-6}$ с.

По формуле (13) определяем среднее время передачи в кольцо пакета данных:

$E[T_p] = \frac{b}{fd} = \frac{800}{2 \cdot 10^6} = 4 \cdot 10^{-4}$ с.

По формулам (11) и (12) определяем загрузки обслуживающего прибора, создаваемые t- и r-очередями соответственно:

$$\rho_t(i) = \lambda_t(i) * E[T_p] = 20 \cdot 4 \cdot 10^{-4} = 0{,}008$$

$$\rho_r(i) = \lambda_r(i) * E[T_p] = 980 \cdot 4 \cdot 10^{-4} = 0{,}392$$

Множество M(70,10) определяется из формулы (7):

M(70,10) = $NS - \{X | (j \le x \le i)\} = \{1{,}2, \dots, 9{,}71{,}72 \dots 100\}$ (т.к. $i \ge j, 70 \ge 10$)

$|M| = 39$

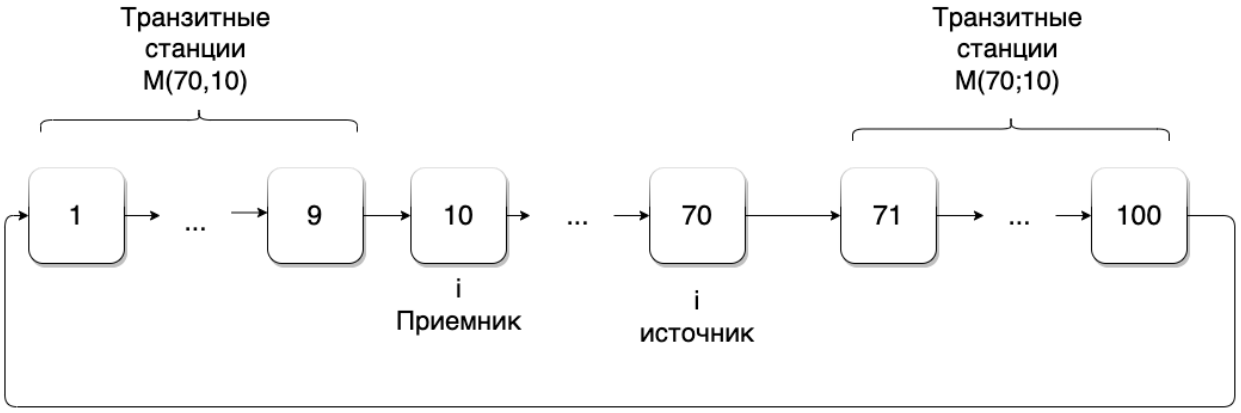

Рисунок 7 – Схема кольцевой ЛВС для задачи нахождения Tf(70,10)

По формуле (9) найдем ожидание начала обслуживания в t-очереди:

$$W_t{}^s(i) = \frac{[\rho_r(i) + \rho_t(i)] * E[T_p{}^2]}{2 * [1 - \rho_r(i)] * E[T_p]} =$$

$$= \frac{[0{,}392 + 0{,}008] * (4 * 10^{-4})^2}{2 * [1 - 0{,}392] * (4 * 10^{-4})} = 13 \cdot 10^{-5} \text{ с.}$$

По формуле (10) найдем ожидание начала обслуживания в r-очереди:

$$W_r{}^{(s)}(i) = d_r^s(i) - W_a(i) = \frac{\rho_t(i) * (1 + p_r(i) * \{1 - p_t(i) - p_r(i)\}) * E[T_p{}^2]}{2 * \frac{[1 - \rho_t(i) - p_r(i)]}{[1 - p_t(i)]} * [1 - p_r(i)] * E[T_p]} =$$

$$\frac{0{,}008 * (1 + 0{,}392 * \{1 - 0{,}008 - 0{,}392\}) * (4 * 10^{-4})^2}{2 * \frac{[1 - 0{,}008 - 0{,}392]}{[1 - 0{,}008]} * [1 - 0{,}392] * (4 * 10^{-4})} = 53 \cdot 10^{-5}.$$

По формуле (6) определяем среднюю величину задержки передачи пакета, сформированного на станции 70 и предназначенного станции 10:

$$T_f(70,10) = W_t(i) + \sum_{n \in M(i,j)} (W_r(n) + \tau_n) + \tau_j + E[T_p] = 13 \cdot 10^{-5} + 39$$

$$* (53 \cdot 10^{-5} + 4{,}05 \cdot 10^{-6}) + 4{,}05 \cdot 10^{-6} + 4 * 10^{-4} = 21{,}3 * 10^{-3}$$

$$= 21{,}3 \text{ (мсек).}$$

Разработка протокола на основе локально-приоритетного доступа и описание используемого формата кадра

Одним из популярных методов подсчета контрольной суммы является CRC. Контрольная сумма - это метод проверки целостности принятой информации на стороне приёмника при передаче по каналам связи. По сути CRC –это результат деления некого объёма информации (информационного сообщения) на константу, а точнее – остаток от деления сообщения на константу. о есть, если хотя бы один бит исходного сообщения изменится при передаче, контрольная сумма тоже изменится, причём существенно. Это большой плюс такой проверки, так как он позволяет однозначно определить, исказилось исходное сообщение при передаче или нет. В качестве константы выбираются числа, кратные байту: 8, 16 или 32.

На стороне источника формируется контрольная сумма CRC, с помощью побитового XOR с выбранным в программной реализации полиномом соответствующей разрядности и записывается в поле КПК.

На стороне приемника при проверке контрольной суммы остаток от деления дополненного сообщения на полином должен равняться нулю-это признак того, что сообщение дошло без ошибок.

В данной работе, при проектировании протокола, необходимо опираться на заданный приоритет. Так как задан приоритет станции, проектируется ЛВС с ***кратным*** *кольцом.*

Допустим, что станция «А» хочет передать данные станции «В», тогда:

1. Станция «А» формирует кадр:

1.1. Формирует признак начала кадра (01111110).

1.2. Формирует заголовок кадра (определяется и присоединяется адрес отправителя и адрес получателя).

1.3. В поле типа кадра заносится 0 – в случае передачи данных и 1 – в случае передачи ответного кадра (кадра с пустым полем данных и отсутствующим полем ДПД).

1.4. В поле данных заносится информация для передачи, а в ДПД количество байт в поле данных.

1.5. Вычисляется контрольная сумма CRC16 и заносится в поле КП. Контрольная сумма охватывает проверку полей заголовка и самой КП.

1.5.1. Создаётся массив (регистр), заполненный нулями, равный по длине разрядности (степени) полинома.

1.5.2. Исходное сообщение дополняется нулями в младших разрядах, в количестве, равном числу разрядов полинома.

1.5.3. В младший разряд регистра заносится один старший бит сообщения, а из старшего разряда регистра выдвигается один бит.

1.5.4. Если выдвинутый бит равен "1", то производится инверсия битов (операция XOR, исключающее ИЛИ) в тех разрядах регистра, которые соответствуют единицам в полиноме.

1.5.5. Если в сообщении ещё есть биты, то вернуться к 1.5.3.

1.5.6. Когда все биты сообщения поступили в регистр и были обработаны этим алгоритмом, в регистре остаётся остаток от деления, который и является контрольной суммой CRC. Содержимое регистра помещается в поле КП.

2. Устанавливается соединение со средой передачи.

3. При получении пакета каждая станция сравнивает адрес получателя со своим адресом. Если они не совпадают, то кадр транспортируется обратно в кольцо, при совпадении адресов управление передается на прием кадра из сети. Станция «В» выполняет следующие действия:

3.1. поиск во входном потоке бит на входном интерфейсе в блоке проверки начала кадра (БПНК) ограничителя начала кадра, после чего начинается прием остальной части кадра через входной интерфейс.

3.2. . Затем в блоке проверки адреса (БПАП) сравнивается поле «адрес получателя» с адресом текущей станции. Если адреса не совпали, то кадр попадает в буфер транзита. Где ожидает своей очереди на транспортировку в выходной интерфейс.

3.3. Если адреса совпали, то осуществляется контроль целостности кадра в блоке (БПО) – блоке проверки ошибок. Алгоритм тот же, что и при формировании контрольной суммы CRC, только в случае приемника в регистре останутся нули, в случае безошибочной передачи.

3.4. Если адреса совпадают и нет ошибок в пакете, то проверяется поле типа пакета в блоке проверки типа пакета (БПТК). Если ТК=0, что соответствует кадру с пользовательской информацией, то считываются данные (переход кадра в буфер приема) и формируется кадр-ответ. Если ТК=1, то пришел кадр-ответ, соответственно кадр с пользовательской информацией был успешно доставлен-обнуляется и останавливается время *timeout*, а буфер передачи очищается.

4. Формирование кадра-ответа станцией «В» (кадр-ответ отправляется после успешной доставки пакета):

4.1. Формирует признак начала кадра (01111110).

4.2. Формирует заголовок кадра (определяется и присоединяется адрес отправителя и адрес получателя).

4.3. В поле типа пакета (ТК) заносится 1.

4.4. Подсчитывается контрольная сумма CRC и заносится в поле КП.

5. Для станции «А» устанавливается $timeout$.(когда сгенерирован кадр, но еще не отправлен) Т.е. если кадр-ответ не приходит в течение $timeout$, то станция «А» принимает решение, что кадр был либо не получен станцией «В», либо содержал ошибки и повторно отправляет его станции «В». Если кадр-ответ получен, то кадр изымается из буфера передачи.

Описание используемого формата кадра

При разработке кадра к нему предъявляются следующие требования:

1) Кадр должен иметь следующие поля: признак начала кадра, адрес получателя (АП), адрес источника (АО), тип сообщения, длину поля данных, поле данных, контрольную последовательность.

2) Кадр должен иметь размер b=800 бит.

Длина кадра с данными меняется в пределах от 54 до 798 бит.

Рассмотрим формат кадра, который выглядит следующим образом.

Таблица 1 – формат кадра с полезной информацией.

1 байт	7 бит	7 бит	1 бит	7 бит	1-94 байт	2 байт
НК	АП	АО	ТК	ДПД	ПД	КП

Биты, задающие режим:

НК – ограничители кадра. Появление этой комбинации бит (01111110) является указанием на то, что следующий байт является адресом получателя.

АП,АО – поле адреса (получателя и отправителя соответственно). Так как в сети 100 станций то 7 бит будет достаточно для представления адреса.

ТК– тип кадра. Тип кадра принимает значение 1, если кадр-ответ и 0, если кадр с информацией.

ДПД – длина поля данных. Определяет количество байт в поле данных.

ПД– поле данных. Сюда записываются данные, которые хочет передать станция. Длина этого поля меняется в пределах от 1 байта до 94 байт. Это поле отсутствует в случае, если формируется кадр-ответ.

КП – контрольная последовательность [все поля кроме начала кадра]. Используется для обнаружения ошибок передачи между двумя станциями. Передающая станция вычисляет КП и включает его в состав кадра. В свою очередь принимающая станция производит аналогичные вычисления над принятым кадром. В результате, в проверяющем регистре должны остаться нули, что будет соответствовать корректно переданному пакету.

Таблица 2 – формат кадра-ответа.

1 байт	1 байт	1 байт	1 бит	2 байт
НК	АП	АО	ТК	КП

Метод контроля ошибок

Циклический избыточный контроль (Cyclic Redundancy Check, CRC) является в настоящее время наиболее популярным методом контроля в вычислительных сетях (и не только в сетях, например, этот метод широко применяется при записи данных на диски и дискеты). Метод основан на рассмотрении исходных данных в виде одного многоразрядного двоичного числа, то есть k-битовой последовательности, где каждый бит является коэффициентом, равным 0 или 1, полинома G(x) степени k. Например, кадр стандарта Ethernet, состоящий из 1024 байт, будет рассматриваться как одно число, состоящее из 8192 бит. В качестве контрольной информации рассматривается остаток R(x) от деления этого числа на известный делитель – порождающий полином P(x). Обычно в качестве делителя выбирается 17- или 33-разрядное число, чтобы остаток от деления имел длину 16 разрядов (2 байта в кадре HDLC) или 32 разряда (4 байта в кадре Ethernet). При получении кадра данных снова вычисляется остаток от деления на тот же делитель P(x), но при этом к данным кадра добавляется и содержащаяся в нем контрольная сумма. Если остаток от деления на R(x) равен нулю, то делается вывод об отсутствии ошибок в полученном кадре, в противном случае кадр считается искаженным. Существует несколько модифицированная процедура вычисления остатка, приводящая к получению в случае отсутствия ошибок известного ненулевого остатка, что является более надежным показателем корректности.

Правило вычисления циклического кода CRC на стороне приемника.

Приемник выполняет деление последовательности k+n битов принятого кадра на тот же самый порождающий полином P(x). Принятое кодированное сообщение с возможной ошибкой можно представить в виде H(x) = F(x) + E(x), где E(x) – полином с ненулевыми коэффициентами в каждой искаженной позиции. Если результат деления равен нулю, то есть E(x) = 0, то считается, что передача кадра выполнена без ошибок. Теоретически может случится так, что сам полином E(x) так же делится на P(x) без остатка. Но и в этом случае ничего не остается, как принять решение, что H(x) не содержит ошибок. Ошибочная комбинация может быть обнаружена, если E(x) не делится на P(x). Именно поэтому порождающий полином P(x) выбирается в зависимости от вида ошибок. Рекомендация МККТТ V41 определяет полином P(x) с n =16 для использования в линиях связи в виде: $P(x) = 1 + x^5 + x^{12} + x^{16}$.

Модификация правила вычисления циклического кода CRC на стороне приемника.

На практике в сетевых контроллерах применяется аппаратная реализация операции деления на полином и, как правило, осуществляется на сдвиговом регистре с соответствующими обратными связями и сумматорами по mod 2. При этом, если не произошло ошибок, то в сдвиговом регистре после обработки принятого кадра вместе с полем КПК будут содержаться только нули. Эта же нулевая последовательность

является исходной для начала приема и проверки следующего кадра. Проблема возникает, если между двумя следующими друг за другом кадрами в силу сбоя пропадает флаг – ограничитель кадра. Тогда оба кадра будут приняты как один правильный кадр. Причина – начальная и конечная битовая комбинация в сдвиговом регистре при правильном приеме совпадают и равны нулю. Поэтому было введено правило для протокола HDLC, по которому начальная и конечная битовые комбинации в сдвиговом регистре должны быть различными. Для этого начальная комбинация изменяется – в сдвиговом регистре перед операцией деления на полином все разряды устанавливаются в единицу. В результате при приеме осуществляется деление многочлена вида: $x^{16}\times G(x)+x^{k}\times L(x)$, где k – степень исходного полинома G(x), а L(x) – дополнительный член, который определяется как

$$L(x) = \sum_{n=0}^{15} x^n$$

Первое слагаемое делится на P(x) без остатка, если нет ошибок. Остаток получается от деления постоянного второго слагаемого и имеет вид:

$$1 + x + x^2 + x^3 + x^8 + x^{10} + x^{12}.$$

Таким образом, остаток от деления отличен от нуля и равен 1111.0000.1010.1000 . Этот метод обладает более высокой вычислительной сложностью, но его диагностические возможности гораздо выше, чем у методов контроля по паритету. Метод CRC обнаруживает все одиночные ошибки, двойные ошибки и ошибки в нечетном числе бит.

Структурная схема сетевого микроконтроллера

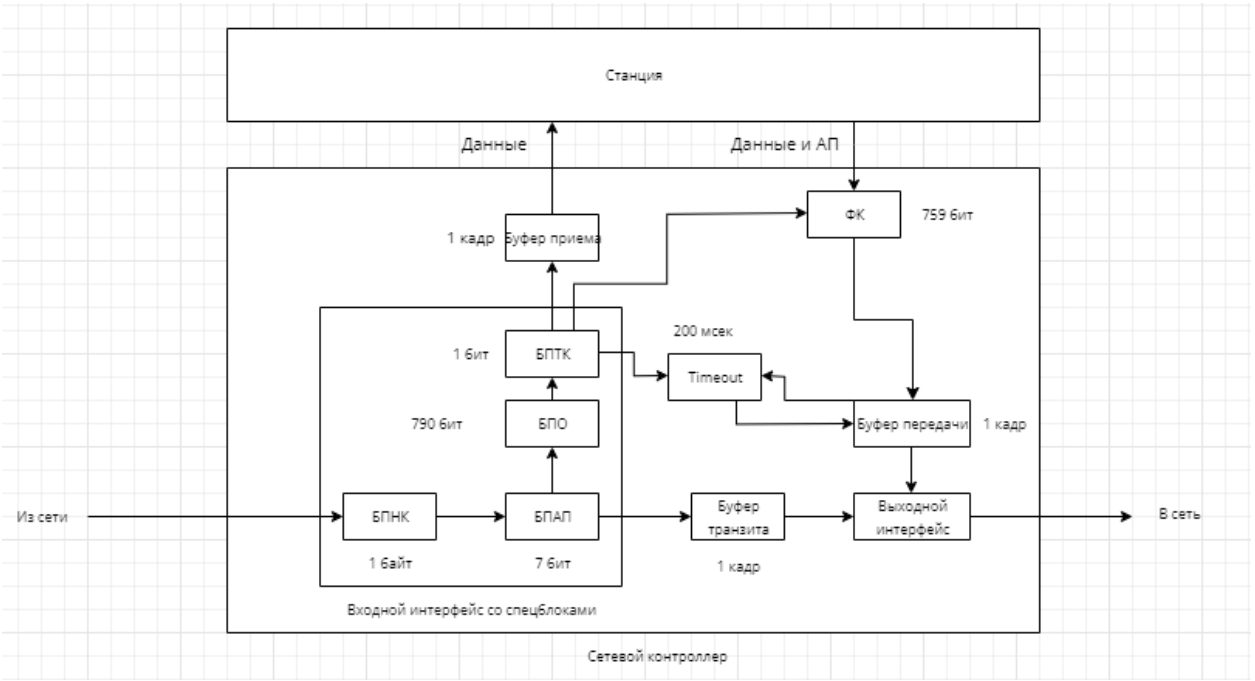

Рисунок 8 – Структура сетевого контроллера

- Составим требования для каждого блока:
- Формирование кадра (ФК). Данный блок получает на вход от станции данные и адрес получателя, после чего формирует заголовок, контрольную последовательность кадра и флаг начала кадра, после чего отправляет готовый кадр в буфер передачи. Данный блок также работает в режиме принятия сигнала от БПТК: в случае, когда пришел кадр с полезной информацией, необходимо сформировать

кадр-ответ, для этого из БПТК присылается адрес источника, все остальные поля будут сформированы внутри блока ФК.

- Для того чтобы станция могла обнаружить начало кадра каждой станции необходимо добавить для каждой станции во входной интерфейс блок начала кадра (БПНК), который просматривает входящий поток бит, если встречается определенная последовательность бит – 01111110, значит следующий байт – это первый байт заголовка кадра, в частности адрес получателя. Объём для этого блока определим в 1 байт.
- Для входного интерфейса Необходим блок, производящий сравнение адреса текущей станции с адресом получателя из кадра (БПАП).В сети 100 станций, поэтому 7 бит будет вполне достаточно. Если адреса совпали, то кадр будет отправлен в БПО. Если кадр был передан некорректно, то кольцо будет очищено от такого кадра. Если адреса не совпали, то кадр попадает в буфер транзита;
- Для буфера приема определим вместимость буфера передачи. Производительность станции неизвестна, поэтому возьмем это значение как максимальный размер кадра, а именно 800 бит;
- Буфер транзита. При приоритете станции буфер транзита должен быть достаточно большим, так как может возникнуть ситуация, в которой станция будет долго передавать пакеты, в то время как транзитные кадры приходят на кольцевой интерфейс. Если пропускная способность кольца 10 Мбит/с, то биты передаются каждые 1/10 мкс. При скорости распространения сигнала порядка $2*10^8$ м/с каждый бит занимает $2*10^8/10*10^6 = 20$ метров в кольце. Одновременно в кольце может находиться 1000/20 = 50 бит. Всего в кадре 800 бит, т.е. одновременно в кольце может находиться лишь часть кадра. Так как, в теории, в каждый момент времени в кратном кольце может находится N кадров и для каждой станции имеется свой буфер транзита, примем размер буфера транзита равным 1 кадр.
- В случае с приоритетом станции, отсутствует необходимость делать буфер передачи слишком большим, так как кадры от станции имеют больший приоритет. Примем за размер буфера размер одного кадра – 800 бит;
- Время для блока timeout необходимо рассчитывать исходя из характеристик сети и количества станций. В нашем случае, 100 станций и необходимо найти среднюю задержку передачи кадра между двумя наиболее отдаленными станциями. $T_f(1{,}100) = W_t(i) + \sum_{n\in M(i,j)}(W_r(n) + \tau_n) + \tau_j + E[T_p] = 13 \cdot 10^{-5} + 99 * (53 \cdot 10^{-5} + 4{,}05 \cdot 10^{-6}) + 4{,}05 \cdot 10^{-6} + 4 * 10^{-4} = 53{,}4 * 10^{-3} = 53{,}4$ (мсек). Для избегания дублирования умножим результат на 4, что в итоге дает примерно 200 мсек.
- Для блока проверки типа пакета необходим 1 бит от контрольного поля. При ТК=1 таймер обнуляется и останавливается, буфер передачи освобождается.
- В выходной интерфейс сетевой контроллер, исходя из приоритета станции, сначала должен передавать кадры из буфера передачи, если такие есть, а только потом из буфера транзита.

Граф-схема состояний сетевого микроконтроллера

Можно определить три состояния, в котором может находиться сетевой контроллер:

- Приемник (R). В этом состоянии СК прослушивает канал, принимает данные в блок входного интерфейса, выбирает и принимает из него то, что адресовано ему.
- Транзит (T). В этом состоянии СК дожидается освобождения выходного интерфейса и передает транзитный кадр, записанный в буфер транзита, далее по кольцу через блок выходного интерфейса в соответствии заданным приоритетом.
- Передатчик (S). В этом состоянии СК дожидается освобождения выходного интерфейса и передает далее по кольцу кадр из выходного буфера, записанный ранее станцией.

Следует учитывать, что при приоритете станции применяется процедура кратного кольца, то есть кадр снимается с кольца получателем и каждая станция в данный момент времени может отправить кадр.

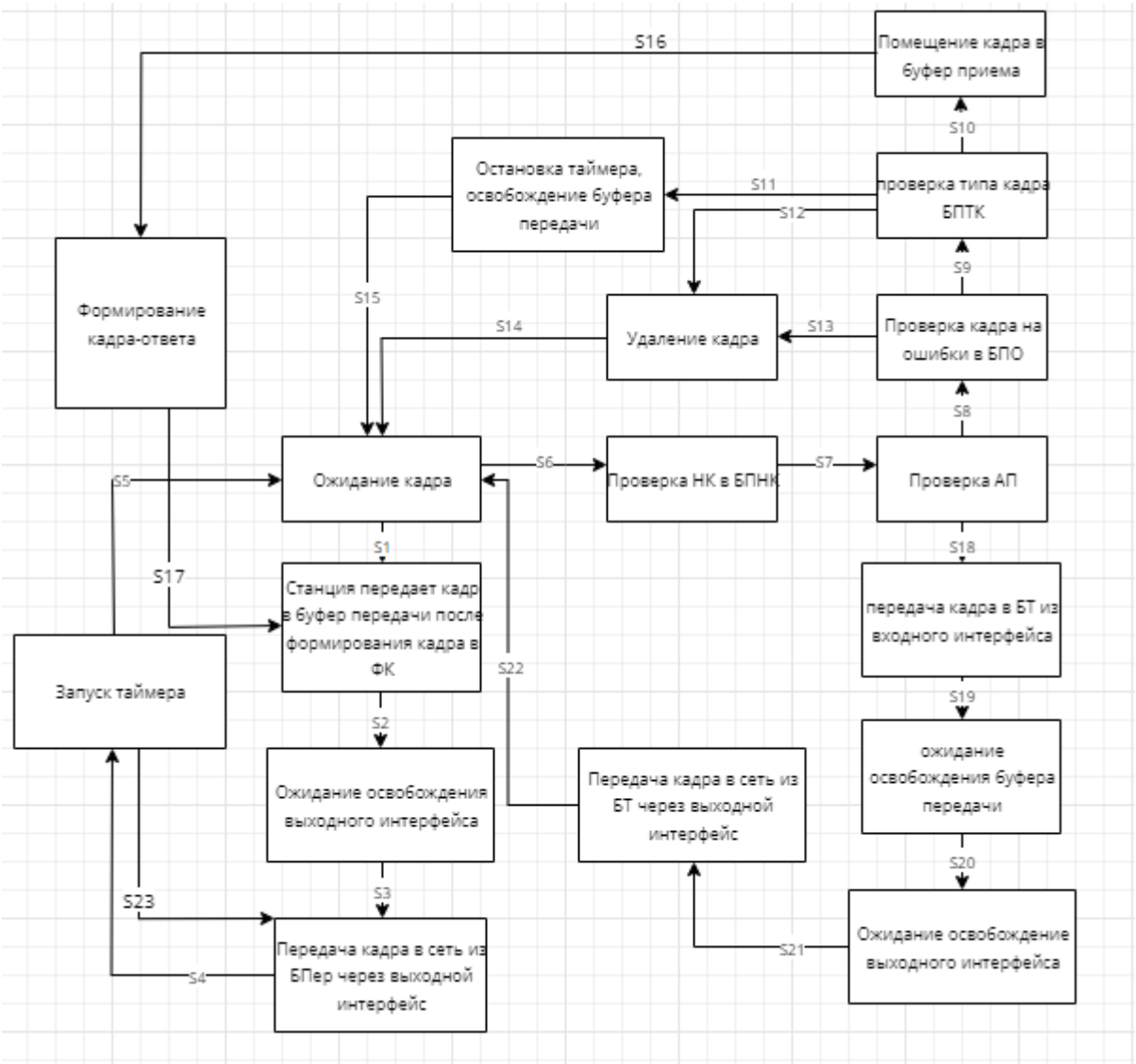

Рисунок 9 – Граф состояний сетевого контроллера

Рассматриваемые переходы:

- Переход **s1** срабатывает при поступлении кадра от текущей станции.
- Переход **s2** срабатывает при помещении кадра от станции в буфер передачи, либо при срабатывании таймера.
- Переход **s3**, **s21** срабатывает при освобождении выходного интерфейса.
- Переход **s4**, **s22** срабатывает при завершении передачи кадра в сеть.

- Переход **s_5** срабатывает после инициализации таймера.
- Переход **s_6** срабатывает при поступлении начала кадра из сети.
- Переход **s_7** срабатывает при поступлении первого байта заголовка-Адреса Получателя.
- Переход **s_8** срабатывает при совпадении адреса станции и адреса получателя.
- Переход **s_9** срабатывает при отсутствии в кадре ошибок.
- Переход **s_{10}** срабатывает, если пришедший кадр является кадром с полезной информацией.
- Переход **s_{11}**, s_{12} срабатывают, если пришедший кадр является кадром-ответом.
- Переход s_{13} срабатывает, если кадр пришел с ошибкой.
- Переход s_{14} срабатывает при удалении предыдущего кадра и начале ожидания нового кадра.
- Переход s_{15} срабатывает, когда кадр-ответ обработан, таймер остановлен и обнулен, буфер передачи освобожден.
- Переход s_{16} срабатывает, когда пришедший кадр с пользовательской информацией принят и обработан.
- Переход s_{17} срабатывает, когда буфер транзита свободен и готов принять кадр-ответ
- Переход s_{18} срабатывает, когда не совпали адрес станции и адрес приемника.
- Переход s_{19} срабатывает, когда данные в буфере-транзите готовы.
- Переход s_{20} срабатывает, когда буфер передачи пуст.

Граф-схема алгоритма процедуры создания и отправки кадра

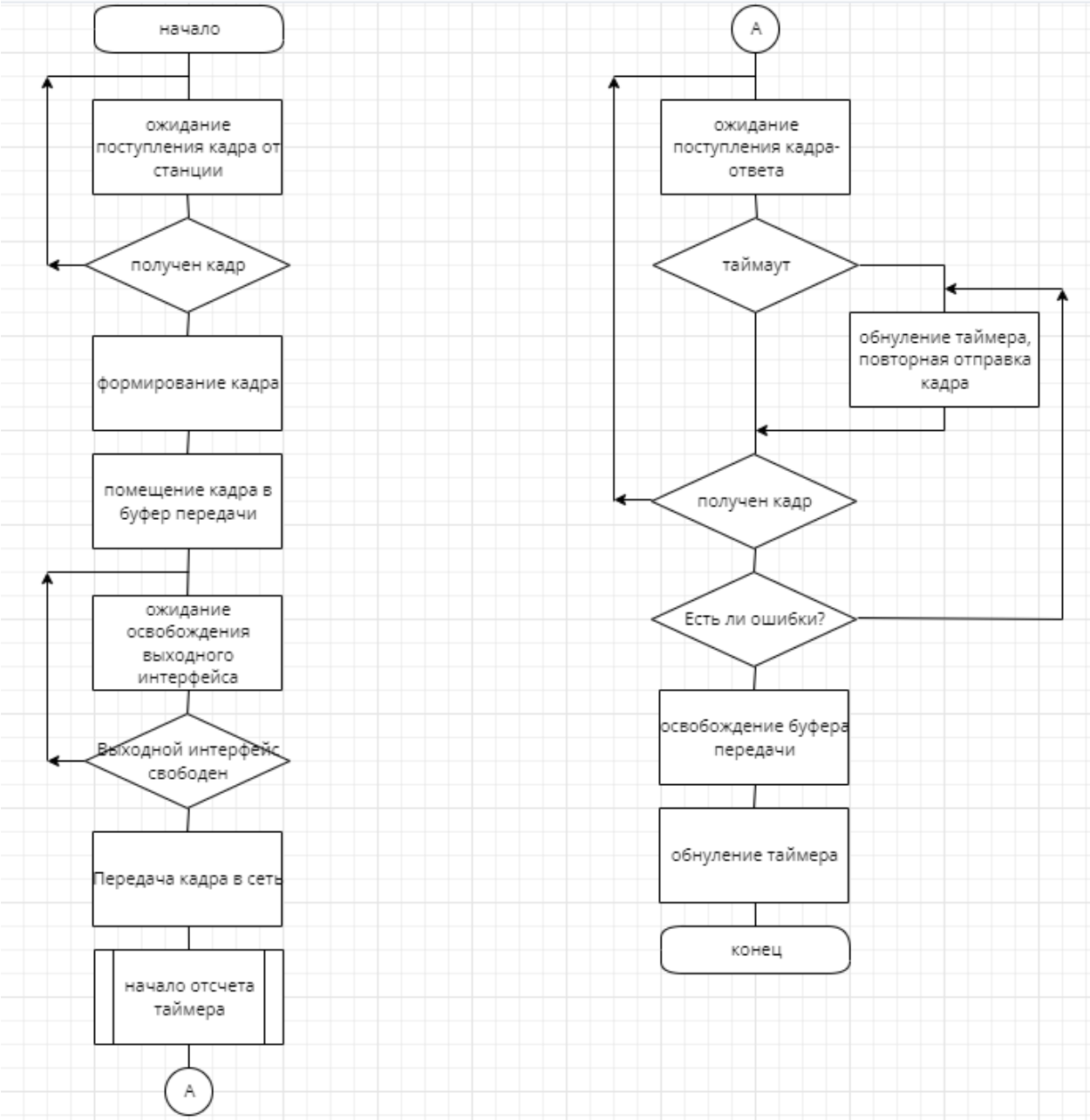

Рисунок 10 – Блок-схема станции-источника

Скриншоты работы программы моделирования

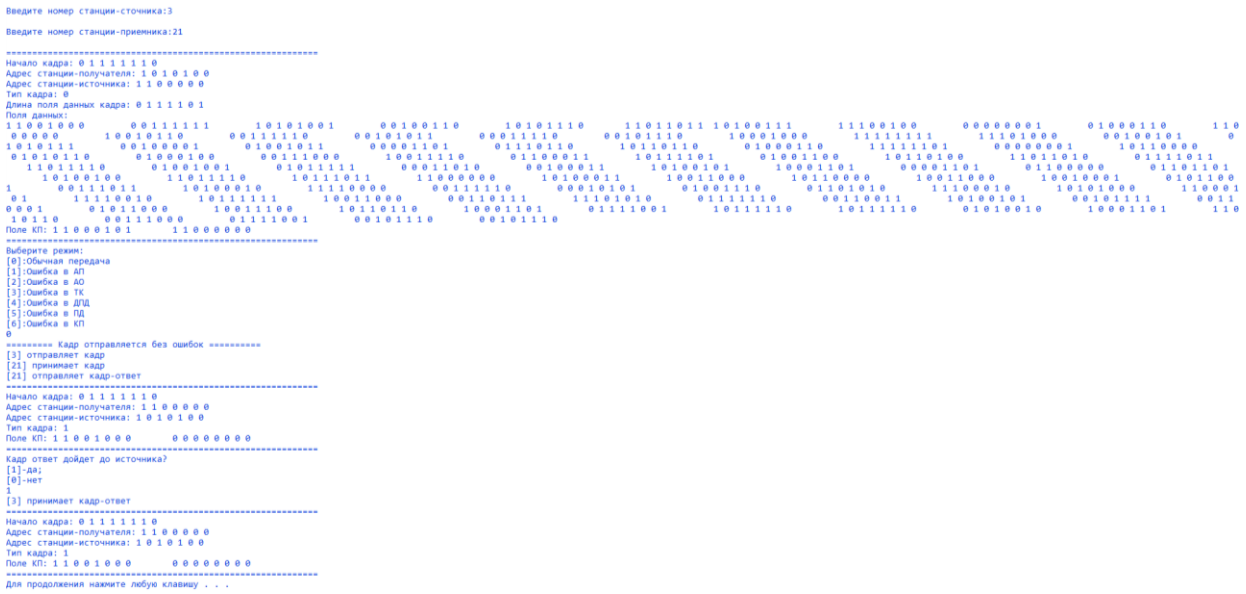

Рис. 11–пример удачной передачи кадра.

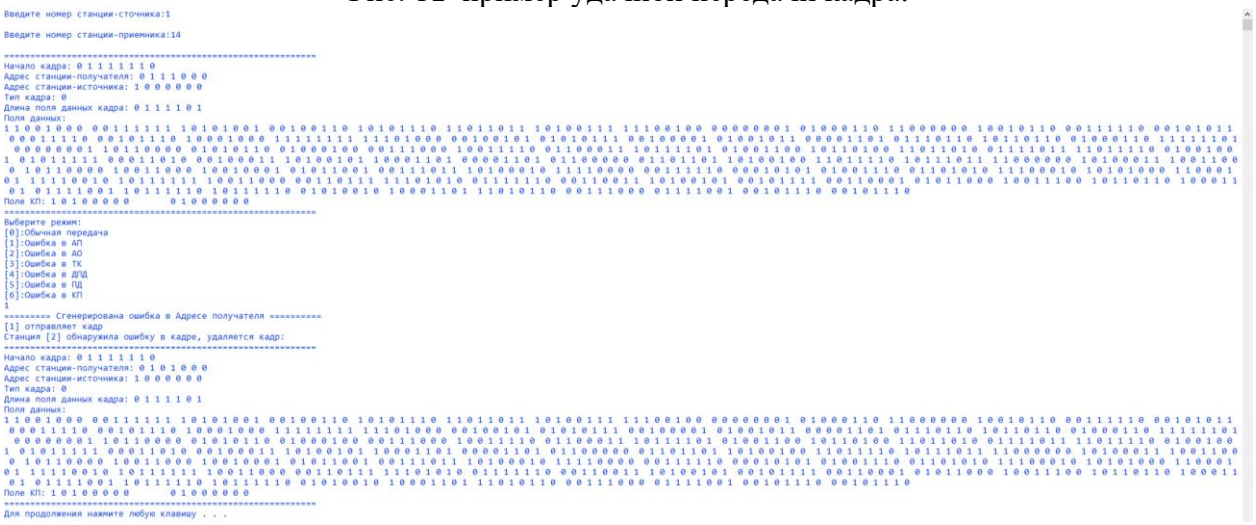

Рис. 12–пример неудачной передачи кадра. Сгенерирована ошибка в поле адреса получателя. Ошибка обнаружена. Кадр удален.

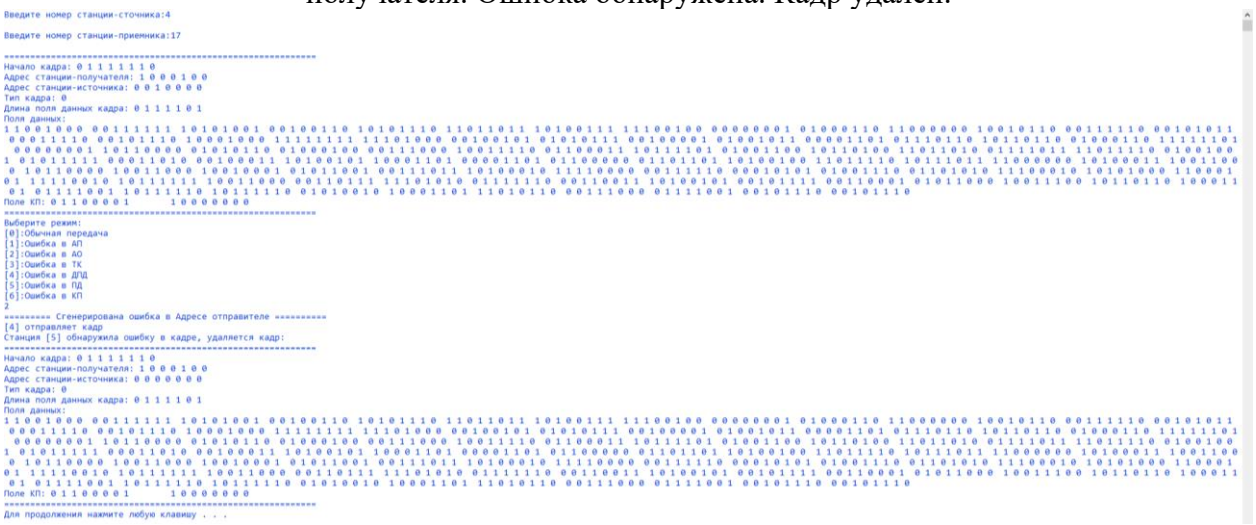

Рис. 13–пример неудачной передачи кадра. Сгенерирована ошибка в поле адреса отправителя. Ошибка обнаружена. Кадр удален.

Рис. 13–пример неудачной передачи кадра. Сгенерирована ошибка в поле типа кадра. Ошибка обнаружена. Кадр удален.

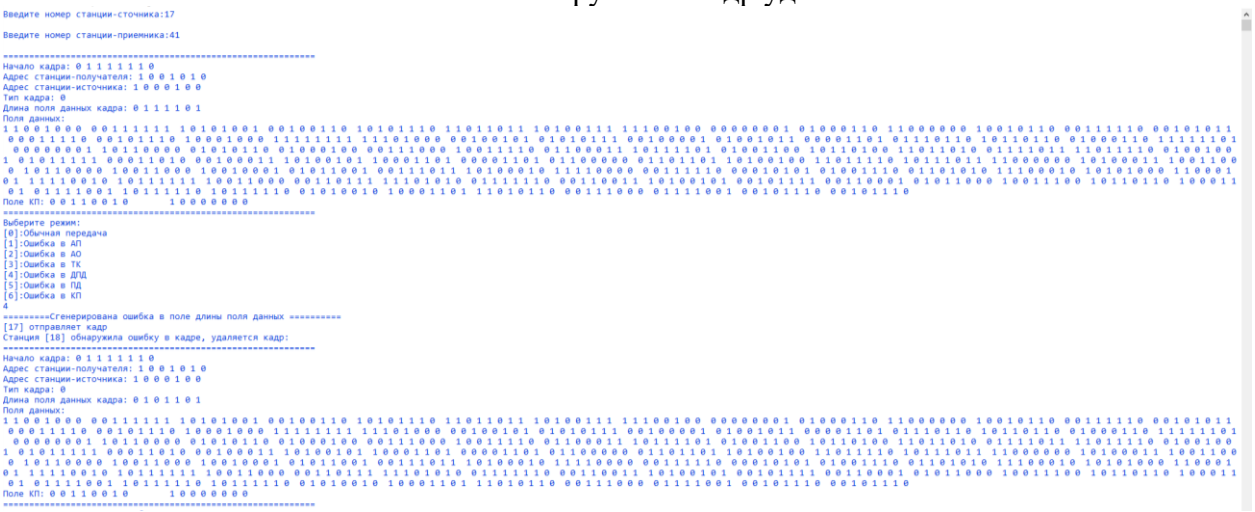

Рис. 14–пример неудачной передачи кадра. Сгенерирована ошибка в поле длины поля данных. Ошибка обнаружена. Кадр удален.

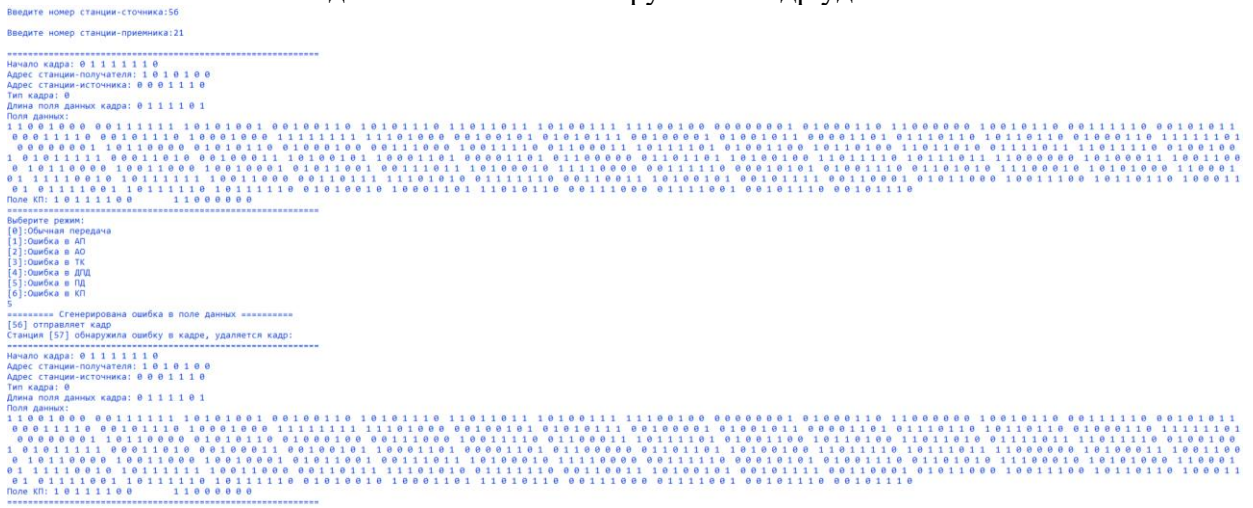

Рис. 15–пример неудачной передачи кадра. Сгенерирована ошибка в поле данных. Ошибка обнаружена. Кадр удален.

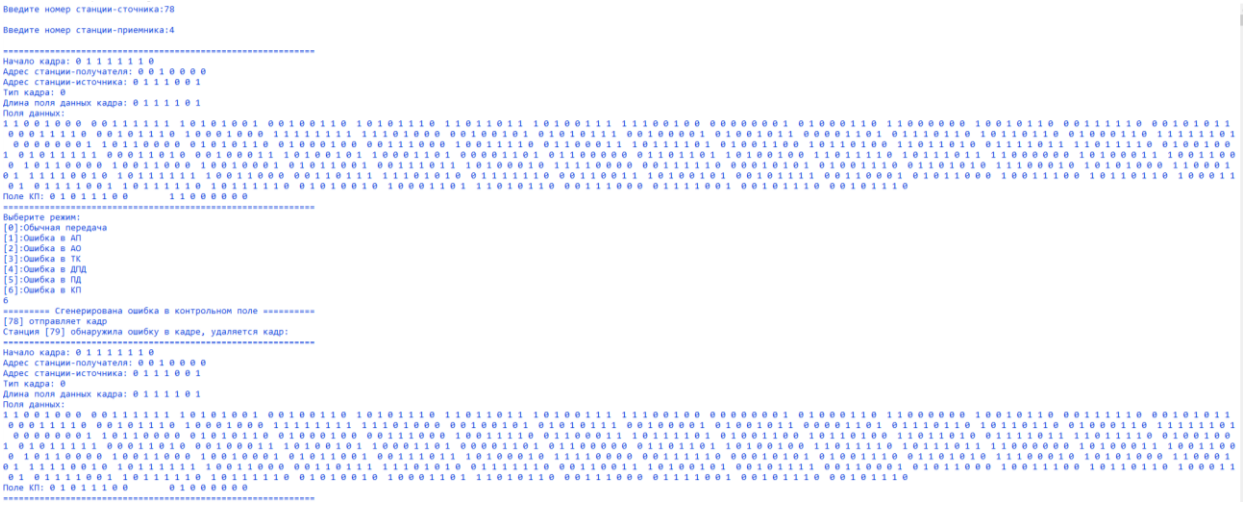

Рис. 16–пример неудачной передачи кадра. Сгенерирована ошибка в контрольном поле. Ошибка обнаружена. Кадр удален.

Рис. 16–пример неудачной передачи кадра-ответа. Ошибка обнаружена. Кадр-ответ удален, из-за чего кадр с информацией отправляется вновь.

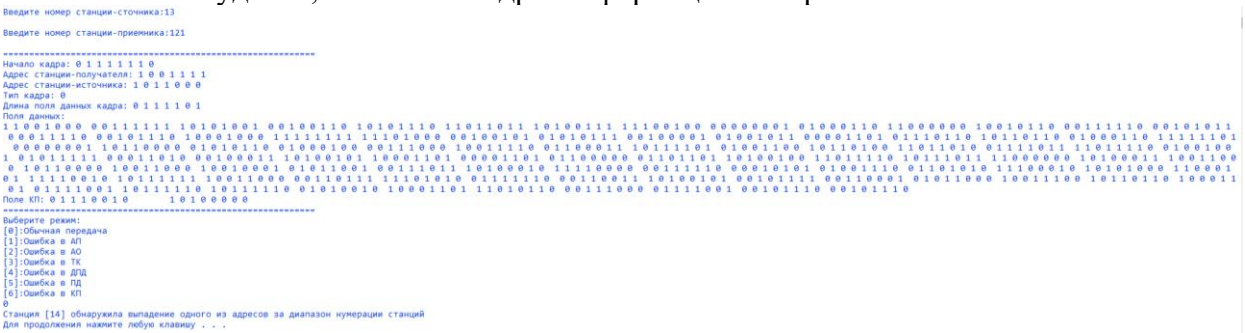

Рис. 17–пример неудачной передачи кадра-ответа. В этом случае был указан адрес, выходящий за пределы диапазона нумерации станций. Ошибка была обнаружена, а кадр удален.

Листинг программы
Main.cpp

```
#include<iostream>
#include<bitset>
#define N  16                    //разрядность полинома
#define SIZE_OF_ADDRES 7         //count of byte * sizeof(byte) for Addres field
#define SIZE_OF_DATA 94*8          //count of byte * sizeof(byte) for DATA field
#define SIZE_OF_FCS 2*8            //count of byte * sizeof(byte) for FCS field
#define SIZE_OF_FLAG 1*8           //count of byte * sizeof(byte) for FLAG field
#define SIZE_OF_TYPE 1        //count of byte * sizeof(byte) for Control field
#define SIZE_OF_LDATA 7        //count of byte * sizeof(byte) for Control field
#define count 100              //количество станций в кольце
const int size_of_inf_package = SIZE_OF_FLAG + 2 * SIZE_OF_ADDRES + SIZE_OF_TYPE + SIZE_OF_LDATA +
SIZE_OF_DATA + SIZE_OF_FCS;
const int size_of_ans_package = SIZE_OF_FLAG + 2 * SIZE_OF_ADDRES + SIZE_OF_TYPE + SIZE_OF_FCS;
char CRC[] = { 0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,1,0,0,0,1,1,1,0,1,1,0,1,1,0,1,1,1 };//двоичное представление полинома
0x04C11DB7
char* package;
char* answer;
using namespace std;
//=======================================Station
Functions=========================================
void print(char* package, bool type);
char* GetPackage(char* DAddr, char* SAddr, char* ldata);
void SetCRC(char* FCS, char* Data, bool type);
bool CheckFCS(char* package, bool type);
char* GetAnswer(char* DAddr, char* SAddr);
int main()
{

   unsigned int source = 0;
   unsigned int dest = 0;
   char AD[SIZE_OF_ADDRES];
   char AS[SIZE_OF_ADDRES];
   char Ldata[SIZE_OF_LDATA];
         setlocale(LC_ALL, "Russian");
   begin:
   cout << "Введите номер станции-сточника:"; cin >> source; cout << endl;
   cout << "Введите номер станции-приемника:"; cin >> dest; cout << endl;
   bitset<SIZE_OF_ADDRES> addrS = { source };
   for (int i = 0; i < SIZE_OF_ADDRES; i++) {
      AS[i] = addrS[i];
   }
   bitset<SIZE_OF_ADDRES> addrD = { dest };
   for (int i = 0; i < SIZE_OF_ADDRES; i++) {
      AD[i] = addrD[i];
   }
   bitset<SIZE_OF_LDATA> ldata = { SIZE_OF_DATA/8 };
   for (int i = 0; i < SIZE_OF_LDATA; i++) {
      Ldata[i] = ldata[i];
   }
   package = new char[size_of_inf_package];
   package = GetPackage(AD, AS, Ldata);
   print(package, 0);
   cout << "Выберите режим:" << endl;
   cout << "[0]:Обычная передача" << endl;
   cout << "[1]:Ошибка в АП" << endl;
   cout << "[2]:Ошибка в АО" << endl;
   cout << "[3]:Ошибка в ТК" << endl;
   cout << "[4]:Ошибка в ДПД" << endl;
   cout << "[5]:Ошибка в ПД" << endl;
   cout << "[6]:Ошибка в КП" << endl;
   int change;
   scanf_s("%d", &change);
```

```
    if (change < 0 || change>6) {
        cerr << "Выход за диапазон выбора режима" << endl; goto begin;
    }
    if (source > count || dest > count)
    {
        cerr << "Станция [" << source + 1 << "] обнаружила выпадение одного из адресов за диапазон нумерации
станций" << endl;
        system("pause");
        return 0;
    }

    switch(change)
    {
    case 0: {//успешная передача
        cout << "========= Кадр отправляется без ошибок ==========" << endl;

        break;
    }
    case 1: {//Ошибка в АП
        cout << "========= Сгенерирована ошибка в Адресе получателя ==========" << endl;
        int errbit = rand() % SIZE_OF_ADDRES + SIZE_OF_FLAG;
        package[errbit] = (package[errbit] == 1) ? 0 : 1;

        break;
    }
    case 2: {//Ошибка в АО
        cout << "========= Сгенерирована ошибка в Адресе отправителе ==========" << endl;
        int errbit = rand() % SIZE_OF_ADDRES + SIZE_OF_ADDRES + SIZE_OF_FLAG;
        package[errbit] = (package[errbit] == 1) ? 0 : 1;

        break;
    }
    case 3: {//Ошибка в ТК
        cout << "========= Сгенерирована ошибка в типе кадра ==========" << endl;
        package[2 * SIZE_OF_FLAG + SIZE_OF_FLAG] = (package[2 * SIZE_OF_FLAG + SIZE_OF_FLAG] == 1) ? 0 :
1;

        break;
    }
    case 4: {//Ошибка в ДПД
        cout << "=========Сгенерирована ошибка в поле длины поля данных ==========" << endl;
        int errbit = rand() % SIZE_OF_LDATA + 2*SIZE_OF_ADDRES + SIZE_OF_FLAG+SIZE_OF_TYPE;
        package[errbit] = (package[errbit] == 1) ? 0 : 1;

        break;
    }
    case 5: {//Ошибка в ПД
        cout << "========= Сгенерирована ошибка в поле данных ==========" << endl;
        int errbit = rand() % SIZE_OF_DATA + 2*SIZE_OF_ADDRES + SIZE_OF_FLAG+
SIZE_OF_LDATA+SIZE_OF_TYPE;
        package[errbit] = (package[errbit] == 1) ? 0 : 1;

        break;
    }
    case 6: {//Ошибка в КП
        cout << "========= Сгенерирована ошибка в контрольном поле ==========" << endl;
        int errbit = rand() % SIZE_OF_FCS + SIZE_OF_DATA + 2 * SIZE_OF_ADDRES + SIZE_OF_FLAG +
SIZE_OF_LDATA + SIZE_OF_TYPE;
        package[errbit] = (package[errbit] == 1) ? 0 : 1;

        break;
    }
    }
    point:
    cout << "["; printf("%d",source); cout << "] отправляет кадр" << endl;
    if (CheckFCS(package, 0)) {
```

```
        cerr << "Станция ["; printf("%d", source + 1); cout << "] обнаружила ошибку в кадре, удаляется кадр:" << endl;
        print(package, 0);
        system("pause");
        return 0;
    }

    cout << "["; printf("%d", dest); cout << "] принимает кадр" << endl;

    answer = new char[size_of_ans_package];
    answer = GetAnswer(AS, AD);
    cout << "["; printf("%d", dest); cout << "] отправляет кадр-ответ" << endl;
    print(answer, 1);
    cout << "Кадр ответ дойдет до источника?"<<endl<<"[1]-да;"<<endl<<"[0]-нет"<<endl;
    scanf_s("%d", &change);
    if (change == 0) {
        cout << "Станция ["; printf("%d", dest+1); cout << "] обнаружила ошибку в кадре-ответе и удалила его" << endl;
        goto point;
    }

    cout << "["; printf("%d", source); cout << "] принимает кадр-ответ" << endl;
    print(answer, 1);
    system("pause");
    delete[]package;
    return 0;
}


//=======================================Station
Functions=========================================
void print(char* package, bool type) {
    int iter = 0;
    if (!type) {

        cout << "============================================================" << endl;
        cout << "Начало кадра: "; for (int i = 0; i < SIZE_OF_FLAG; i++)  { printf("%d ", package[i]); } cout << endl;
        iter = SIZE_OF_FLAG;
        cout << "Адрес станции-получателя: "; for (int i = 0; i < SIZE_OF_ADDRES; i++) { printf("%d ", package[iter]);
iter++; }cout << endl;
        cout << "Адрес станции-источника: ";  for (int i = 0; i < SIZE_OF_ADDRES; i++) { printf("%d ", package[iter]);
iter++; }cout << endl;
        cout << "Тип кадра: ";  printf("%d\n", package[iter]); iter++;
        cout << "Длина поля данных кадра: "; for (int i = 0; i < SIZE_OF_LDATA; i++) { printf("%d ", package[iter]);
iter++; }cout << endl;
        cout << "Поля данных: " << endl; for (int i = 0; i < SIZE_OF_DATA; i++) { printf("%d ", package[iter]); if ((i + 1)
% 8 == 0)cout << '\t'; iter++; }cout << endl;
        cout << "Поле КП: "; for (int i = 0; i < SIZE_OF_FCS; i++) { printf("%d ", package[iter]); if ((i + 1) % 8 == 0)cout
<< '\t'; iter++; }cout << endl;
        cout << "============================================================" << endl;
    }
    else
    {
        cout << "============================================================" << endl;
        cout << "Начало кадра: "; for (int i = 0; i < SIZE_OF_FLAG; i++) { printf("%d ", package[i]); } cout << endl;
        iter = SIZE_OF_FLAG;
        cout << "Адрес станции-получателя: "; for (int i = 0; i < SIZE_OF_ADDRES; i++) { printf("%d ", package[iter]);
iter++; }cout << endl;
        cout << "Адрес станции-источника: ";  for (int i = 0; i < SIZE_OF_ADDRES; i++) { printf("%d ", package[iter]);
iter++; }cout << endl;
        cout << "Тип кадра: ";  printf("%d \n", package[iter]); iter++;
        cout << "Поле КП: "; for (int i = 0; i < SIZE_OF_FCS; i++) { printf("%d ", package[iter]); if ((i + 1) % 8 == 0)cout
<< '\t'; iter++; }cout << endl;
        cout << "============================================================" << endl;
    }
}
char* GetAnswer(char* DAddr, char* SAddr) {
    char Flag[SIZE_OF_FLAG] = { 0,1,1,1,1,1,1,0 };
```

```
    char FCS[SIZE_OF_FCS];

    for (int i = 0; i < SIZE_OF_FCS; i++) { FCS[i] = 0; }
    char* buf = new char[size_of_ans_package - 8];
    int iter = 0;
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { buf[iter] = DAddr[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { buf[iter] = SAddr[i]; iter++; }
    buf[iter] = 0; iter++;
    for (int i = 0; i < SIZE_OF_FCS; i++) { buf[iter] = 0; iter++; }

    SetCRC(FCS, buf, 1);
    for (int i = 0; i < SIZE_OF_FCS; i++) {
        buf[2 * SIZE_OF_ADDRES + SIZE_OF_TYPE + i] = FCS[i];
    }
    iter = 0;
    for (int i = 0; i < SIZE_OF_FLAG; i++) { answer[iter] = Flag[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { answer[iter] = DAddr[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { answer[iter] = SAddr[i]; iter++; }
    answer[iter] = 1; iter++;
    for (int i = 0; i < SIZE_OF_FCS; i++) { answer[iter] = FCS[i]; iter++; }
    delete[]buf;
    return answer;
}
char* GetPackage(char* DAddr, char* SAddr, char* ldata) {
    char Flag[SIZE_OF_FLAG] = { 0,1,1,1,1,1,1,0 };
    char* data = new char[SIZE_OF_DATA];
    char FCS[SIZE_OF_FCS];

    for (int i = 0; i < SIZE_OF_DATA; i++) { data[i] = (char)round(rand() % 2); }
    for (int i = 0; i < SIZE_OF_FCS; i++) { FCS[i] = 0; }
    char* buf = new char[size_of_inf_package-8];
    int iter = 0;
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { buf[iter] = DAddr[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { buf[iter] = SAddr[i]; iter++; }
    buf[iter] = 0; iter++;
    for (int i = 0; i < SIZE_OF_LDATA; i++) { buf[iter] = ldata[i]; iter++; }
    for (int i = 0; i < SIZE_OF_DATA; i++) { buf[iter] = data[i]; iter++; }
    for (int i = 0; i < SIZE_OF_FCS; i++) { buf[iter] = 0; iter++; }

    SetCRC(FCS, buf, 0);
    for (int i = 0; i < SIZE_OF_FCS; i++) {
        buf[2 * SIZE_OF_ADDRES + SIZE_OF_TYPE+ SIZE_OF_LDATA + SIZE_OF_DATA + i] = FCS[i];
    }
    iter = 0;
    for (int i = 0; i < SIZE_OF_FLAG; i++) { package[iter] = Flag[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { package[iter] = DAddr[i]; iter++; }
    for (int i = 0; i < SIZE_OF_ADDRES; i++) { package[iter] = SAddr[i]; iter++; }
    package[iter] = 0; iter++;
    for (int i = 0; i < SIZE_OF_LDATA; i++) { package[iter] = ldata[i]; iter++; }
    for (int i = 0; i < SIZE_OF_DATA; i++) { package[iter] = data[i]; iter++; }
    for (int i = 0; i < SIZE_OF_FCS; i++) { package[iter] = FCS[i]; iter++; }

    delete[]buf;
    return package;
}
void SetCRC(char* FCS, char* Data, bool type) {
    int c = 0; int buf[N]; int in;
    int size_data;
    if (type == 0) { size_data = size_of_inf_package-8; }
    else { size_data = size_of_ans_package-8; }
    for (int i = 0; i < N; i++)buf[i] = 0;
    for (int i = 0; i < size_data; i++) {
        c = buf[N - 1];//получение смещенного числа
        in = Data[i];//постановка нового числа из сообщения
        for (int j = N - 1; j > 0; j--) { buf[j] = (c & CRC[j]) ^ buf[j - 1]; }//XOR
        buf[0] = (c & CRC[0]) ^ in;
```

```
    }
    for (int j = 0; j < N; j++) { FCS[j] = buf[N - 1 - j]; }
}
bool CheckFCS(char* package, bool type) {
    int c = 0; int buf[N]; int in;
    int size_data;
    if (type == 0) { size_data = size_of_inf_package - 8; }
    else { size_data = size_of_ans_package - 8; }
    for (int i = 0; i < N; i++)buf[i] = 0;
    for (int i = 0; i < size_data; i++) {
        c = buf[N - 1];//получение смещенного числа
        in = package[SIZE_OF_FLAG + i];//постановка нового числа из сообщения
        for (int j = N - 1; j > 0; j--) { buf[j] = (c & CRC[j]) ^ buf[j - 1]; }//XOR
        buf[0] = (c & CRC[0]) ^ in;
    }
    for (int j = 0; j < N; j++) {
        if (buf[j] == 1)return true;
    }
    return false; //знак того, что нет ошибок
}
```

Вывод

В ходе выполнения курсового проекта были рассчитаны средняя задержка передачи кадра по сети и вероятность прохождения транзитом через определенный узел, разработаны алгоритм контроля ошибок, протокол достоверной передачи данных и реализована программная модель работы станции в сети.

Моделирование работы станции в сети при заданных условиях и ограничениях выполнено. Программа работает корректно.

Содержание полей обычного (информационного) кадра и кадра-ответа представлено в таблице 1 и таблице 2. Формат кадра описан корректно.

Разработанные алгоритм контроля ошибок и протокол достоверной передачи кадра позволяют получить результат, удовлетворительный для поставленной задачи. Однако, протокол имеет свои преимущества и недостатки.

Из преимуществ можно выделить высокую надежность протокола: изменение одного бита проверяемых полей повлечет за собой сильную разницу от требуемого результата.

Недостатком протокола является высокая продолжительность подсчета контрольной суммы или ее проверки.