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 Герасимец А. В. группа № 1842
8 февраля 2022 г. Срок исполнения 20 апреля 2022 г.

1. ТЕХНИЧЕСКИЕ УСЛОВИЯ

Вариант 5.

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 = 90. 7. Интенсивность входного потока λ_t = 15 1/с. 8. Длина кадра b = 900 бит. 9. Приоритет кольца. 10. Длина сети L = 2 км. 11. Метод – Odd/Even. 12. Алгоритм децентрализованного (транзитного) мониторного узла (Монитор Дц). 13. Определение вероятности прохождения кадра от станции №80 до станции №50. 14. Определение задержки между станциями №60 и №50.

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 _____ /А.В. Герасимец/
(подпись)

Содержание

1. Обоснование соответствия модели кольца со вставными регистрами принципам работы ЛВС с детерминированным доступом

1.1. Методы доступа в ЛВС

В ЛВС можно выделить две основные группы вариантов доступа к моноканалу: случайные и детерминированные методы. В ЛВС с шинной топологией широко используется случайный доступ, а в кольцевых топологиях – детерминированный.

1.2. Детерминированные методы доступа

При методах детерминированного доступа все компьютеры через свои кольцевые адаптеры (станции) осуществляют согласованное взаимодействие с моноканалом в виде кольца. Детерминированный доступ основан на поочередном предоставлении абонентским системам разрешения на передачу кадра. Различают три подхода к реализации методов детерминированного доступа к моноканалу:

1. пропорциональный доступ;
2. приоритетного доступа;
3. локально – приоритетный доступ.

Пропорциональный доступ исключает возможность конфликта источников за счет задания очередности доступа абонентов к каналу. Продвижение очередности производится либо синхросигналами, либо признаками (маркерами) разрешения доступа.

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:

- сегментированное кольцо,
- кольцо типа Cambridge Ring (CR),
- с переменной задержкой,
- кольцо 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.

1.3. Кольцевые ЛВС со вставными регистрами. Структура кольцевого адаптера.

В кольцах со вставными регистрами конфликты между кадрами, находящимися у станций и готовыми к передаче, и кадрами, идущими по кольцу, динамически разрешаются путем включения последовательных сдвиговых регистров в кольцо. Все регистры включены внутрь адаптеров кольца. Вставные регистры выполняют функцию буферов кадров. На рис. 1 показана структура кольцевого адаптера, в котором кроме буферов передачи и приема кадров, включен вставной буфер транзита.

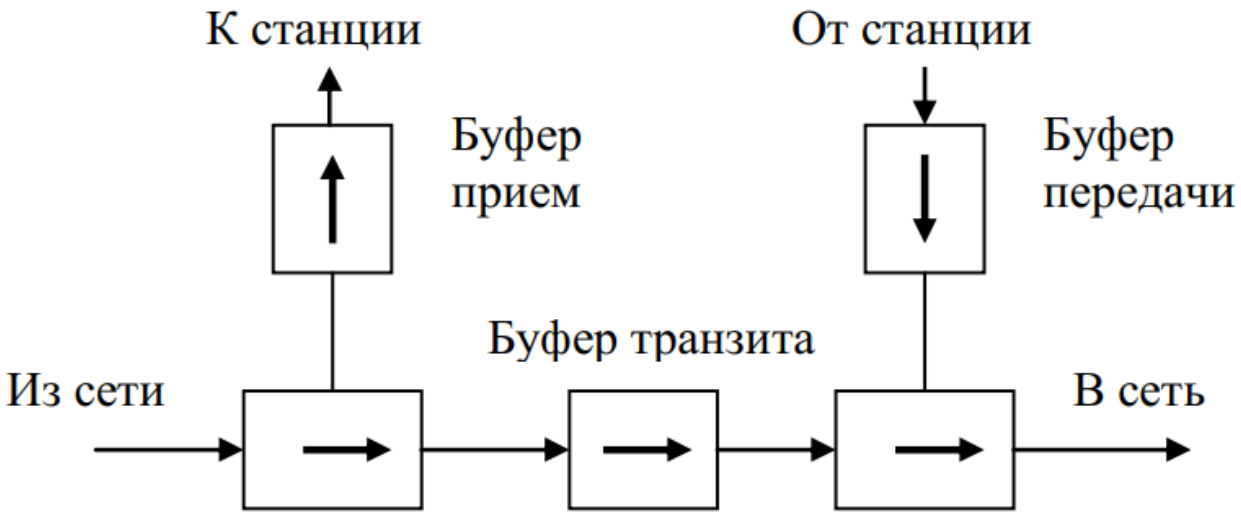

Рисунок 1 - Кольцевой адаптер со вставным буфером транзита

ЛВС со вставными регистрами – кольцевая сеть, обеспечивающая однонаправленную передачу данных между станциями по одной физической среде с возвратом данных к передающей станции. *В кольце может применяться процедура передачи кадра с возвратом данных к передающей станции. Эта процедура соответствует простому кольцу.* Если кадр снимается с кольца получателем, то такая процедура будет соответствовать кратному кольцу.

Кадр данных, подготовленный станцией для выдачи в сеть, временно буферизируется в буфере передач кольцевого адаптера. Порядок и начало выдачи кадра из адаптера зависит от принятого способа функционирования сети, то есть от установленной системы приоритетов при передаче данных. В соответствии с локально-приоритетным методом доступа различают *приоритет кольца* и приоритет станции. Во время передачи станцией своих собственных кадров данных в кольцо остальные кадры, приходящие от других станций, но не предназначенные для данной станции, запоминаются в буфере транзита ее кольцевого адаптера. Если же станция не передает свои собственные кадры, то далее по кольцу передается содержимое вставного буфера транзита кольцевого адаптера.

Порядок приема кадра получателем реализуется в блоке входного интерфейса в зависимости от процедуры передачи кадров, соответствующей простому или кратному кольцу. Процедура передачи кадров для простого или кратного кольца выбирается в зависимости от заданного приоритета кольца или станции. При более высоком *приоритете кольца* применяется *процедура простого кольца*. При более высоком приоритете станции применяется процедура кратного кольца.

1.4. Модель кольца с буферами транзита

Модель кольца с буферами транзита при наличии трех станций изображена на рис. 2. Для описания функционирования системы положим, что кадр, сформированный станцией 1 и предназначенный для станции 3, проходит через эту модель.

После генерации кадр помещается в буфер передачи станции 1, где ожидает своей очереди на передачу. Кадры, идущие по кольцу, запоминаются в буфере транзита и вставляются в расписание в соответствии с принятой дисциплиной обслуживания: с учетом чей приоритет выше – *приоритет кольца* или станции. Начало передачи кадра означает, что началось обслуживание и кадр поступает в блок задержки станции 2. Это является важнейшим свойством данной модели.

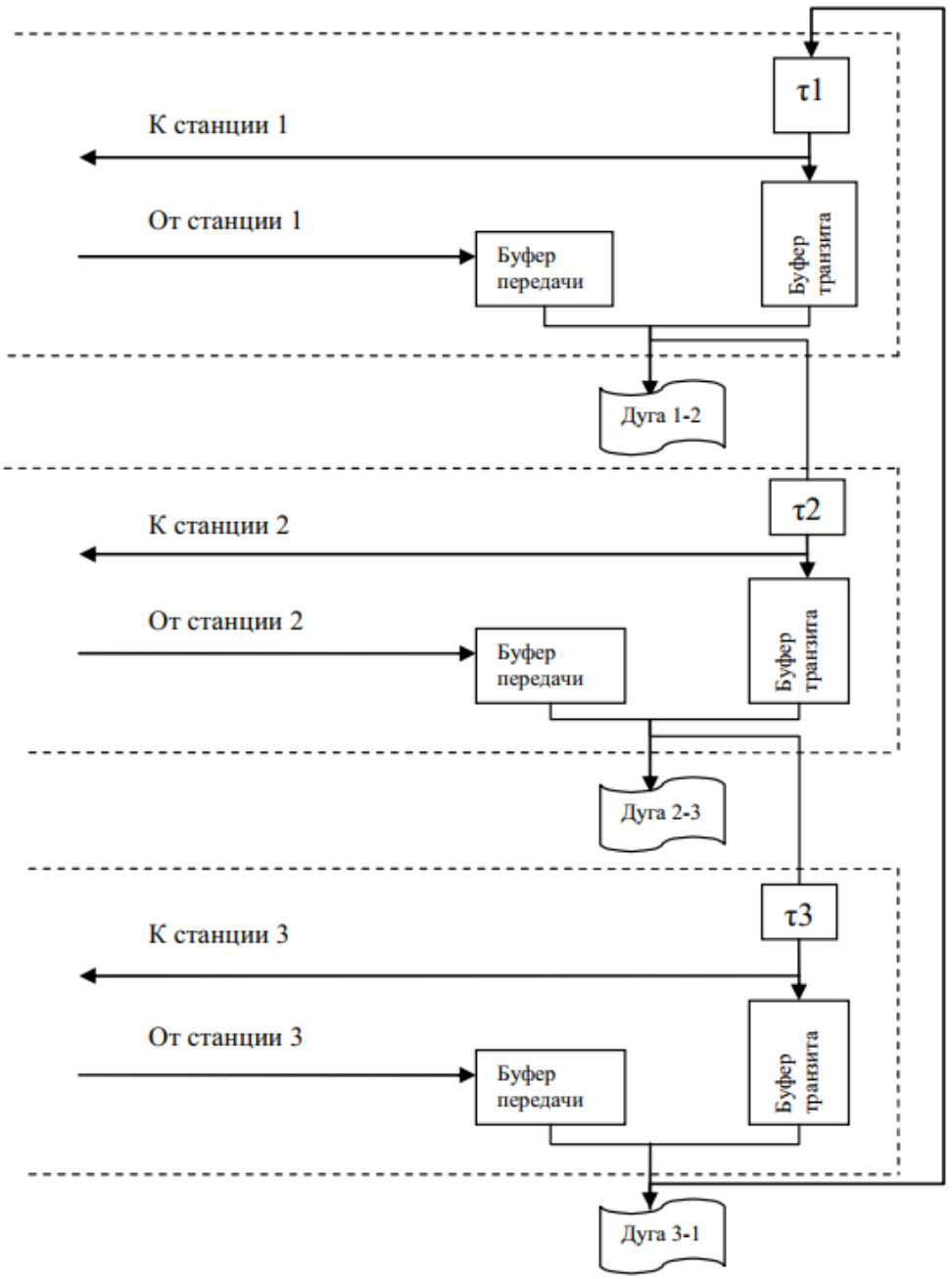

Рисунок 2 - Модель кольца с буферами транзита для трех станций

Блоки задержки τ1, τ2 и τ3 учитывают задержку передачи, необходимую для дешифрации адреса, и время распространения сигнала по соответствующей дуге кольца.

В блоке задержки станции 2 кадр считается задержанным на постоянное время τ2, а затем он помещается в буфер транзита станции 2.

Считается, что пакет поступает в блок задержки станции 3, когда он передается по дуге 2-3.

Проследовав постоянную задержку τ3, кадр поступает к станции 3, т.е. в буфер приема, не обозначенный на рис. 2. При этом кадр покидает модель.

Анализ представленной модели может быть упрощен, если принять допущение, что емкость буфера приема достаточно высока, чтобы не влиять на передачу в кольце потока кадров.

Интерес представляет оценка задержки передачи кадра между двумя станциями, которая представляет собой временной интервал от момента генерации кадра на станции i до его получения на станции j и обозначается как $T_f(i,j)$.

При анализе будут использоваться следующие допущения.

1. Входные потоки кадров от всех станций являются пуассоновскими с интенсивностями $\lambda_t(1), \ldots, \lambda_t(S), \ldots, \lambda_t s(N)$, где S – номер очередной станции, N –общее число станций.
2. Время передачи кадра в кольцо может быть распределено по произвольному закону, а распределение длин кадров, сформированных различными станциями, идентичны.
3. Отношения между источниками и получателями кадров определяются произвольно задаваемой матрицей выбора маршрута $|P(i,\ j)|$, где $P(i,\ j)$ – вероятность того, что кадр, сформированный на станции i, адресован станции j.
4. Задержка τ_i кольцевого адаптера i постоянна. Для простоты в величину τ_i также включается задержка распространения сигналов, передаваемых между адаптерами (i-1) и i.

Анализ представленной модели основан на декомпозиции полной модели кольца в более простую подмодель. Затем производится анализ подмодели и вычисляются значения задержек полной модели на основании результатов, полученных для подмоделей.

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

Анализ подмодели упрощается при использовании вытекающих из сделанных ранее следующих предположений:

- входной поток буфера транзита является пуассоновским;
- время поступления кадра и время обработки, т. е. время его передачи, взаимно независимы.

Действительное время задержки кадра данных определяется суммой трех слагаемых: времени задержки в очереди в буфере передачи, времени передачи, времени суммарных задержек в кольцевых адаптерах кольца.

Необходимо отметить два обстоятельства:

- очередь имеется только в станции-источнике;
- подмодель имеет ненулевую задержку в каждом адаптере кольца, следовательно, сумма всех задержек в очереди зависит от среднего числа кольцевых адаптеров, через которые должен пройти кадр.

Кроме предположений, введенных ранее, дополнительно вводятся следующие:

- момент поступления кадра 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 следующих компонентов: очередей на станции i, дуги $i - (i + 1)$, очередей на станции $(i - 1)$, дуги $(i - 1) - i$. Подмодель станции i показана на рис. 3

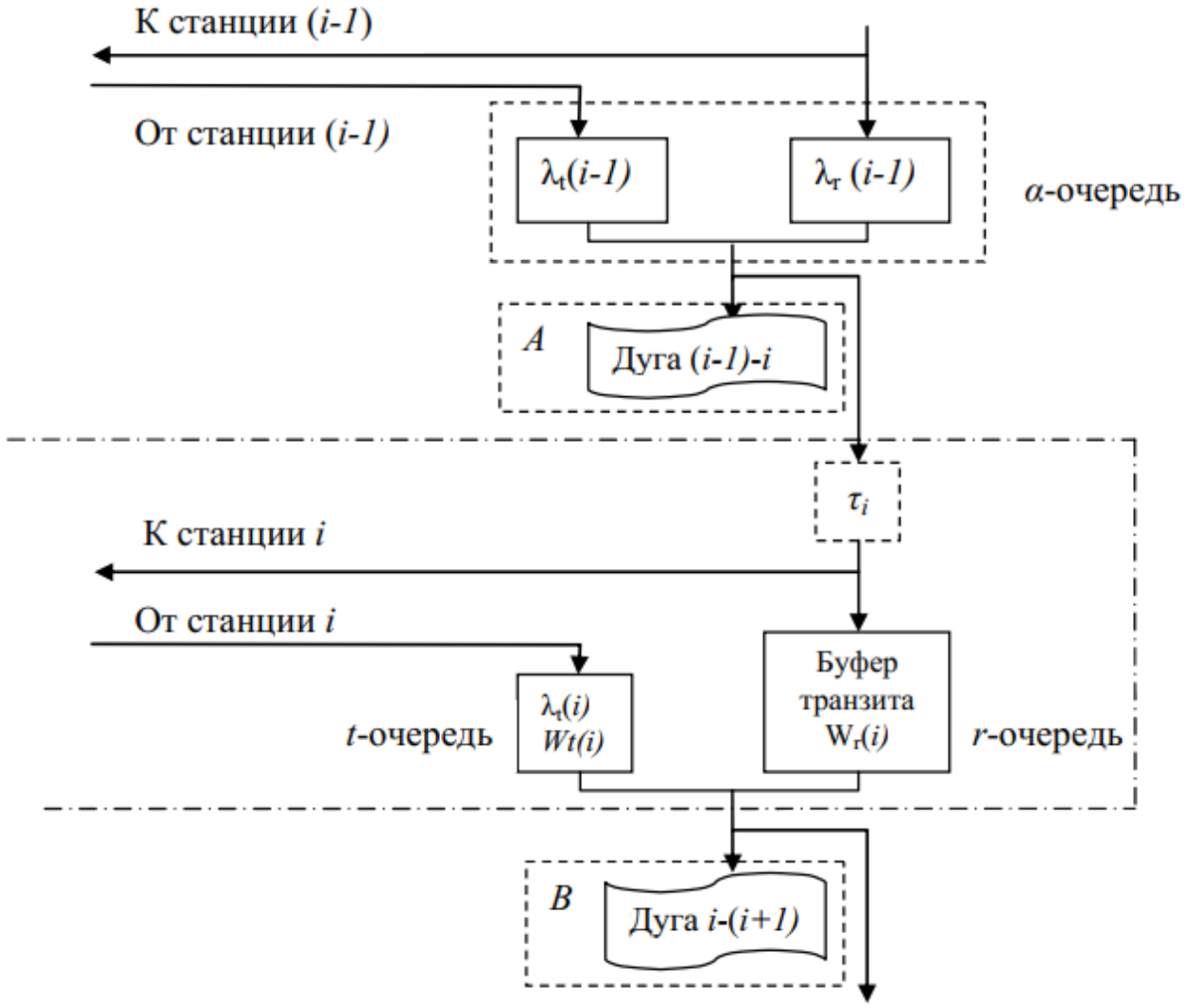

Рисунок 3 - Подмодель станций с очередями

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

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

Для того чтобы определить задержки в подмодели станции i, можно использовать упрощенную интерпретацию этой подмодели (на рис. 3 это сделано с помощью пунктирных рамок). Это упрощение делается в три этапа:

- без потери общности можно считать, что задержка τ_i равна нулю, поскольку задержка приводит просто к временному сдвигу;
- все кадры, поступающие на станцию i, рассматриваются как транзитные кадры;
- входной процесс поступления кадров к входному (фиктивному) буферу станции i не подчиняется действию обычных правил приоритета, применяемых к буферу передачи и буферу транзита станцию (i-1)

Следовательно, можно соединить оба буфера на станции (i-1) в один буфер и организовать одну очередь. В упрощенном варианте подмодели на рис. 3 она обозначена как α-очередь. Объединенный входной поток кадров в очереди будет пуассоновским с интенсивностью $\lambda_r(i)$. Все другие компоненты подмодели станции i являются неизменными в упрощенной подмодели станции i (рис. 3). При этом дуги кольца являются обслуживающими приборами:

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

В упрощенном варианте буфер передачи станции i носит название «t-очередь», буфер транзита – «r-очередь».

Основные свойства упрощенной подмодели станции i следующие:

- кадры, полученные для приема в r-очередь, начинают обслуживаться прибором A;
- время обслуживания кадров в обслуживающих приборах A и B идентично.

Входной поток, поступающий в r-очередь на рис. 3, является составным, так как его слагаемые поступают от буфера передачи и буфера транзита станции (i-1). Суммарная интенсивность этого потока $\lambda_r(i)$.

Необходимо отметить, что наличие прибора A 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 кадры кольца вместо того, чтобы поступать в α-очередь, направляются в прямо к r-очереди, как это показано в «эквивалентной модели» на рис. 4.

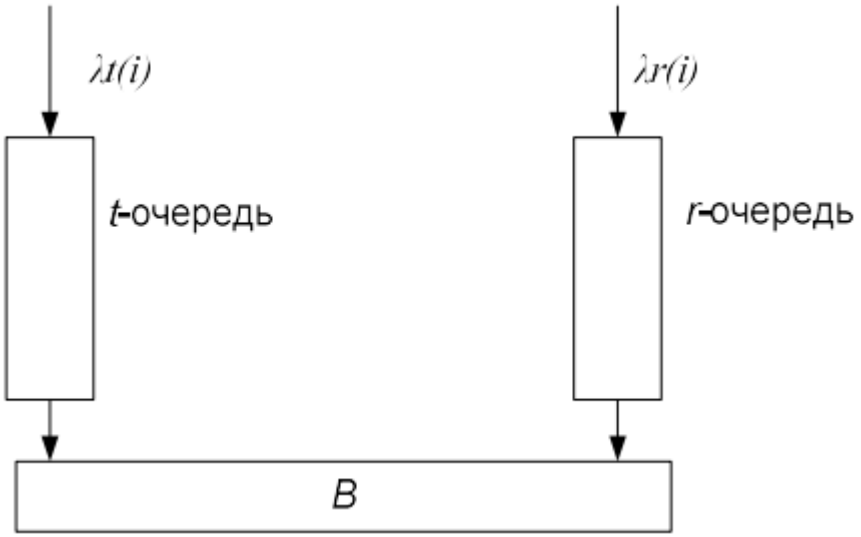

Рисунок 4 - Эквивалентная модель станции

В упрощенном варианте подмодели на рис. 3 обслуживание кадра в приборе A начинается либо намного раньше, чем обслуживание в приборе B (при наличии кадров в очереди r), либо точно в тот же самый момент времени (если нет кадров в очереди r). Следовательно, если допускается, что модели на рис. 3 и 4 управляются идентичными входными потоками,

обслуживающий прибор В (рис. 3) всегда будет обрабатывать те же самые кадры из r- или t-очередей, как и обслуживающий прибор В на рис. 4, если рассматривается начало нового обслуживания, т. е. если нет остаточных кадров в r-очереди.

Дисциплина обслуживания кадров в каждой очереди – FIFO («первый пришел – первый обслужен»). Поэтому процессы обработки в обслуживающем приборе В в подмоделях (рис. 3 и 4) идентичны. Таким образом, задержка $d_r(i)$ от момента поступления кадра в α-очередь (рис.3) до того, как его обслуживание начнется в обслуживающем приборе В, равна задержке в объединенной r-очереди (рис. 4). Отсюда вытекает, что эквивалентная модель станции i – это система массового обслуживания класса M|G|1 (пуассоновский входной поток, произвольное время обслуживания, один обслуживающий прибор), свойства которой хорошо известны.

2. Определение функциональных зависимостей основных характеристик проектируемой ЛВС

Исходные данные:

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 = 90.

7. Интенсивность входного потока $\lambda_t = 15\ 1/\text{с}$.

8. Длина кадра b = 900 бит.

9. Приоритет кольца.

10. Длина сети L = 2 км.

11. Метод – Odd/Even.

12. Алгоритм децентрализованного (транзитного) мониторного узла (Монитор Дц).

13. Определение вероятности прохождения кадра от станции №80 до станции №50.

14. Определение задержки между станциями №60 и №50.

2.1. Теоретическая часть

Для дисциплины обслуживания с относительными приоритетами в случае приоритета кольца (r) ожидание обслуживания в r-очереди будет равно

$$d_r^{(r)}(i) = \frac{[\rho_t(i)+\rho_r(i)]\times E[T_p^2]}{2*[1-\rho_r(i)]\times E[T_p]},$$

где $\rho_t(i) = \lambda_t(i) * E[T_p]$,

$\rho_r(i) = \lambda_r(i) * E[T_p]$.

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

Среднее время передачи в кольцо пакета данных определяется как:

$E[T_p] = \frac{b}{fd}$, где b – длина кадра [бит], fd – скорость передачи кадра [бит/с].

Ожидание начало обслуживания в t-очереди будет следующим:

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

Средняя величина задержек в α-очереди дается следующим выражением:

$$W_\alpha(i) = \frac{\rho_r(i)\times E[T_p^2]}{2*[1-\rho_r(i)]\times E[T_p]}.$$

Следовательно, ожидание обслуживания (передача кадра) для r-очереди будет таким:

$$W_r^{(r)}(i) = d_r^{(r)}(i) - W_\alpha(i) = \frac{\rho_t(i)\times E[T_p^2]}{2*[1-\rho_r(i)]\times E[T_p]}.$$

Для определения интенсивности $\lambda_r(i)$ потока кадров у каждой станции в кольце, содержащем всего N станций, необходимо определить вероятность $Q(i,k)$ того, что кадр, сформированный на станции i, проходит транзитом через станцию k, при этом

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

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

$$I(i,k) = \left\{ \begin{array}{c} M - \{X | i < x \le k\} \\ X | k < x \le i \end{array} \right\}, \begin{array}{l} если\ i < k; \\ если\ i > k; \end{array}$$

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 кадров в буфер транзита может быть определена по формуле

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

Для симметричного потока в системе, содержащей N станций, справедливо:

$$P(i,j) = \frac{1}{N-1}, i \neq j;$$

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

Вероятность передачи кадра из станции i самой себе считаем равной нулю и в случае симметричного потока $\lambda_t(i) = \lambda_t = const$. Тогда

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

Средняя величина задержки передачи кадра, сформированного на

станции i и предназначенного для станции j, определяется следующим выражением:

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

где $W_t(i)$ – среднее время ожидания в t– очереди у передающей станции, то есть в буфере передачи; $\Sigma(W_r(n) + \tau_n)$ – суммарная задержка передачи кадра 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; $E[T_p]$ – среднее время передачи в кольцо кадра данных.

Здесь

$$M(i,j) = \begin{cases} \{X | i < x < j\} \\ NS - \{X | j \le x \le i\} \end{cases}, \begin{array}{l} если\ i < j; \\ если\ i > j; \end{array}$$

где NS - множество станций в кольце, $NS = \{1,\ 2,\ \ldots,\ N\}$.

Величина τ_n находится так:

$$\tau_n = \tau_\alpha + t_{nx} = \frac{\omega}{F_T} + \frac{L/N}{c}$$

где τ_α – активная составляющая задержки, t_{nx} – задержка передачи по каналу, ω – разрядность буферного регистра последовательного интерфейса, FT – тактовая частота сдвига, L – длина сети, N – число станций ЛВС, c=200000 км/с – скорость распространения сигнала по кабелю.

2.2. Расчеты по заданным формулам

Для симметричного потока в системе, содержащей $N = 90$ станций, получаем

$$P(i,j) = \frac{1}{90-1} = \frac{1}{89}$$

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

Тогда:

$$\left[P(i,j)\right]=\begin{bmatrix} 0 & \frac{1}{89} & \frac{1}{89} & \dots & \frac{1}{89} \\ \frac{1}{89} & 0 & \frac{1}{89} & \dots & \frac{1}{89} \\ \frac{1}{89} & \frac{1}{89} & 0 & \dots & \frac{1}{89} \\ \dots & \dots & \dots & \dots & \dots \\ \frac{1}{89} & \frac{1}{89} & \frac{1}{89} & \dots & 0 \end{bmatrix}$$

Необходимо определить вероятность того, что пакет, сформированный на станции 80, проходит транзитом через станцию 50, т.е. Q (80, 50).

Найдем величину I(80, 50) по формуле (2), где M = {1, 2, 3, 4, … 90} и X = {51, 52, 53, 54, … 80}, так как, $i \geq k$ т.е., $80 \geq 50$, то

$I(80,50) = \{X|50 < x \leq 80\} = |\{51,\ 52,\ 53,\ 54,\ \dots 80\}| = 30$

(количество узлов между 50 и 80 узлами).

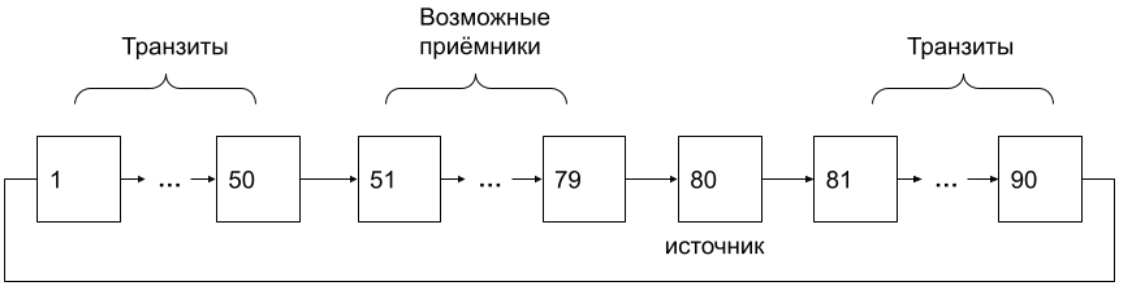

Рисунок 5 - Схема кольцевой ЛВС для задачи нахождения Q(80,50)

По формуле находим Q(80,50):

$Q(80,50) = P(80,51) + P(80,52) \dots + P(80,80)$

$= \frac{1}{89} + \frac{1}{89} + \dots + 0) = \frac{29}{89}$, так как $P(80,80) = 0$

В случае симметричного потока $\lambda_t(i) = 15$ 1/с, тогда с учетом $P(i,i) = 0$, имеем λ_r, находящееся по формуле:

$$\lambda_r = 15 * \frac{1}{90-1} * [\frac{90*90-90}{2} - (90 - 1)] = 660\frac{1}{c}$$

По формуле определяем τ_n:

$$\tau_n = \frac{\omega}{F_T} + \frac{L/N}{c} = \frac{8}{2*10^6} + \frac{2/90}{2*10^5} = 4,11 * 10^{-6}c$$

По формуле определяем среднее время передачи в кольцо пакета данных:

$$E[T_p] = \frac{900}{2*10^6} = 4,5 * 10^{-4}c$$

По формулам определяем загрузки обслуживающего прибора, создаваемые t- и r-очередями соответственно:

$$\rho_t(i) = \lambda_t(i) * E[T_p] = 15 * 4,5 * 10^{-4} = 0,00675$$

$$\rho_r(i) = \lambda_r(i) * E[T_p] = 660 * 4,5 * 10^{-4} = 0,297$$

Множество M(60,50) определяется из формулы:

$M(60,50) = NS - \{X|(j \leq x \leq i)\} = \{1, 2,..., 49, 61, 62,... 90\}$,

так как $i \geq j, 60 \geq 50, \ |M| = 79$

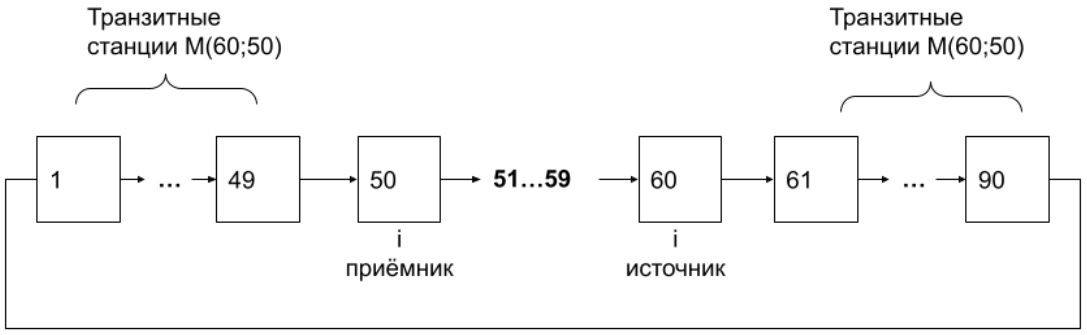

Рисунок 6 - Схема кольцевой ЛВС для задачи нахождения $Tf(60,50)$

По формуле найдем ожидание начала обслуживания в t-очереди:

$$W_t^{(r)}(i) = \frac{[0{,}297+0{,}00675]*(4{,}5*10^{-4})^2}{2*[1-0{,}297-0{,}00675]*[1-0{,}297]\div(4{,}5*10^{-4})} = 0,2758 * 10^{-10} c$$

По формуле найдем ожидание начала обслуживания в r-очереди:

$$W_r^{(r)}(i) = d_r^{(r)}(i) - W_{\alpha}(i) = \frac{0{,}00675*(4{,}5*10^{-4})^2}{2*[1-0{,}297]*(4{,}5*10^{-4})} = 2,162 * 10^{-6} c$$

По формуле определяем среднюю величину задержки передачи пакета, сформированного на станции 60 и предназначенного станции 50:

$$T_f(60,50) = 0,2758 * 10^{-10} + 79 * (2,162 * 10^{-6} + 4,11 * 10^{-6}) +$$

$$+ 4,11 * 10^{-6} + 4,5 * 10^{-4} = 0,9496 \text{ мсек}$$

3. Описание используемого формата кадра и метода контроля ошибок

3.1. Описание используемого формата кадра

Таблица 1 - Используемый формат кадра

8 бит	7 бит	1 бит	7 бит	1 бит	1 бит	1 бит	7 бит	1 бит	7 бит	1 бит	…	7 бит	1 бит
ФН	АП	Р	АО	Р	ТК	БО	ДПД	Р	ПД1	Р	…	ПД108	Р

1. ФН - флаг начала соответствует значению “01111110”, указывает на то, что следующие за ним данные - являются данными адреса получателя
2. Р - бит четности. Рассчитывается для каждого информационного поля. Станция источник выполняет сложение по модулю 2 (mod 2) в каждом информационном поле и заносит в бит четности либо 0, либо 1. После получения кадра рабочая станция выполняет собственное вычисление по модулю 2, сравнивает полученный бит четности с принятым. Если бит четности не совпадает, то станция приемник формирует кадр-ответ с ошибкой.
3. АП - адрес получателя принимают значения от 1 до 90 (0000001 до 1011010) , так как количество станций N = 90
4. АО - адрес отправителя принимают значения от 1 до 90 (0000001 до 1011010) , так как количество станций N = 90
5. ТК - тип кадра принимает значение 0 или 1, где 0 - кадр данных, 1 - кадр ответа.
6. БО - бит ошибки принимает значение 0 или 1, где 0 - нет ошибок, 1 - есть ошибка. Устанавливается источником в «0». Может быть установлен в «1» получателем, либо транзитным узлом. Получателем – в случае, когда при проверке пакета значения битов четности не

соответствуют принятым значениям. Транзитным узлом – в случае, когда в поле «адрес получателя» либо в поле «адрес источника» находится значение, не соответствующее указанному диапазону

7. ДПД - длина поля данных принимает значение от 0 до 108 (0000000 до 1101100)
8. ПД - поле данных содержит данные, которые передает станция.

3.2. Описание метода контроля ошибок

Контроль по паритету представляет собой наиболее простой метод контроля данных. В то же время это наименее мощный алгоритм контроля, так как с его помощью можно обнаружить только одиночные ошибки в проверяемых данных. Метод заключается в суммировании по модулю 2 всех бит контролируемой информации. Например, для данных 100101011 результатом контрольного суммирования будет значение 1. Результат суммирования также представляет собой один бит данных, который пересылается вместе с контролируемой информацией.

При искажении при пересылке любого одного бита исходных данных (или контрольного разряда) результат суммирования будет отличаться от принятого контрольного разряда, что говорит об ошибке. Однако двойная ошибка, например 110101010, будет неверно принята за корректные данные. Поэтому контроль по паритету применяется к небольшим порциям данных, как правило, к каждому байту, что дает коэффициент избыточности для этого метода 1/8.

4. Описание структуры сетевого контроллера и графа его состояний

4.1. Описание структуры сетевого контроллера

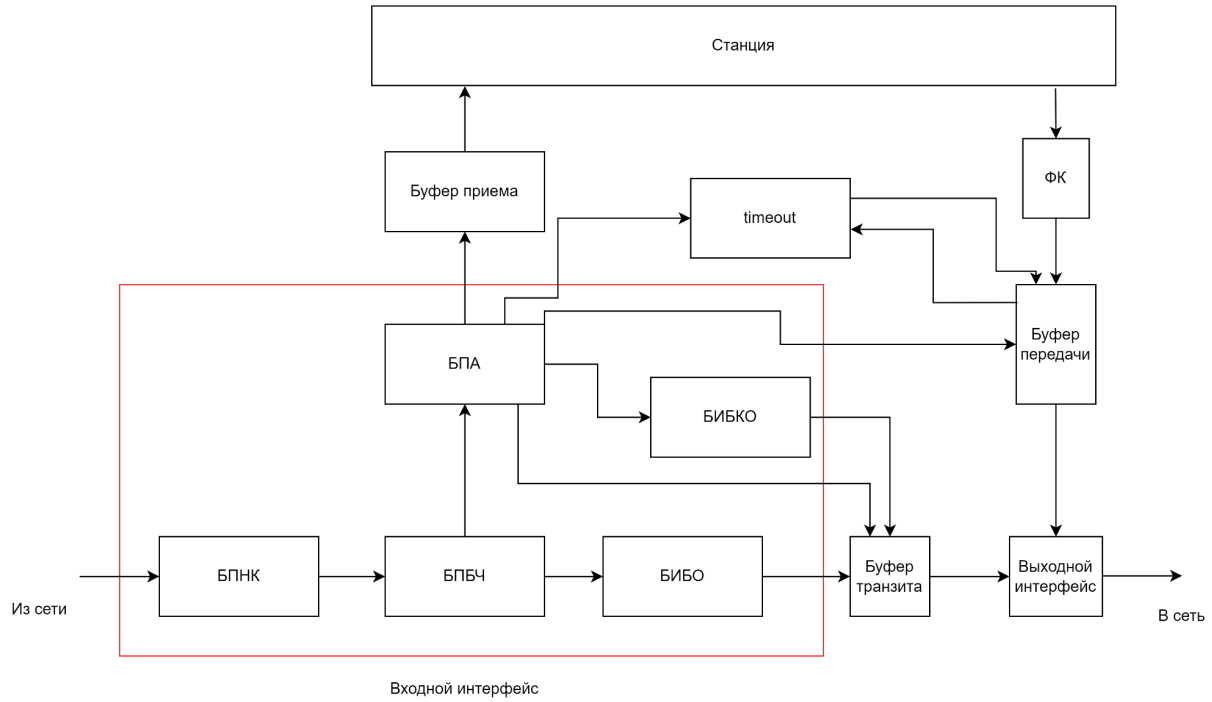

Рисунок 7 - Структура сетевого контроллера

Требования для каждого блока:

- БПНК - блок проверки начала кадра. Добавляется для каждой станции, чтобы определить первые восемь бит, если эти биты 01111110, то из этого следует, что следующие 7 бит являются данными адреса получателя.
- БПБЧ - необходим для контроля ошибок в кадре. Проверяется бит четности с помощью сложения значений по модулю два, затем сравнивается со значением бита четности: если совпадает, то ошибки нет, если не совпадает - кадр искажен.

- БПА - блок проверки адреса. Текущая станция сравнивает адреса получателя и отправителя со своим. при совпадении адреса у текущей станции с адресом отправителя и при это тип кадра равен 1, то сбрасывается timeout и буфер передачи очищается.
- Буфер приема - определяется значением 900 бит (размер кадра), так как производительность станции неизвестна.
- БИБО - блок инициализации бита ошибки принимает значение 0 или 1, где 0 - нет ошибок, 1 - есть ошибка.
- БИБКО - блок инициализации бита кадра ответа. В поле тип кадра будет 1, так как кадр ответа.
- Буфер транзита - для хранения 1 кадра объем 900 бит. При приоритете кольца объем буфера транзита не должен быть большим.
- Буфер передачи - станция не передает кадр в буфер передачи, если он еще не полный, поэтому объем у буфера передачи для хранения 1 кадра 900 бит.
- ФК - формирователь кадра, объем 9 бит. На данном этапе задаются значения полям адреса получателя, отправителя, типа кадров, поля наличия ошибок, вычисляются биты четности. Так как данные идут от станции, то в поле тип кадра значение 0 - кадр данных.
- Так как приоритет кольца, то сетевой контроллер в выходной интерфейс передает при наличии кадры из буфера транзита, а затем из буфера передачи.

Время для блока timeout необходимо рассчитывать исходя из характеристик сети и количества станций. В нашем случае для 90 станций и необходимо найти среднюю задержку передачи кадра между двумя наиболее отдаленными станциями

Timeout вычисляется по формуле:

$$Timeout = T_f(1, 90) + T_f(90, 1) + t_{\text{форм}} + t_{\text{обработки}}$$

Рассчитаем время равное средней величине задержки для передачи кадра на наиболее отдалённую станцию:

$$T_f(1, 90) = 0,2758 * 10^{-10} + 89 * (2,162 * 10^{-6} + 4,11 * 10^{-6}) + 4,11 * 10^{-6} +$$

$$+ 4,5 * 10^{-4} = 1,01 \text{ мсек}$$

$$T_f(90, 1) = 0,2758 * 10^{-10} + (2,162 * 10^{-6} + 4,11 * 10^{-6}) + 4,11 * 10^{-6} +$$

$$+ 4,5 * 10^{-4} = 0,46 \text{ мсек}$$

$t_{\text{форм}}, t_{\text{обработки}}$ берем 0,5 каждый

Тогда:

$$Timeout = 1,01 + 0,46 + 0,5 + 0,5 = 2,47 \text{ мс}$$

Полученное время необходимо брать в десятикратном размере, так как могут быть станции, которые будут работать медленно, что приведет к нежелательному дублированию, соответственно timeout=20(мсек)

4.2. Граф-схема состояний сетевого контроллера

В кольце может применяется процедура передачи кадра с возвратом данных к передающей станции. Эта процедура соответствует простому кольцу.

Необходимо определить и описать детальный граф состояний сетевого контроллера для заданного режима работы. Можно определить 3 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.

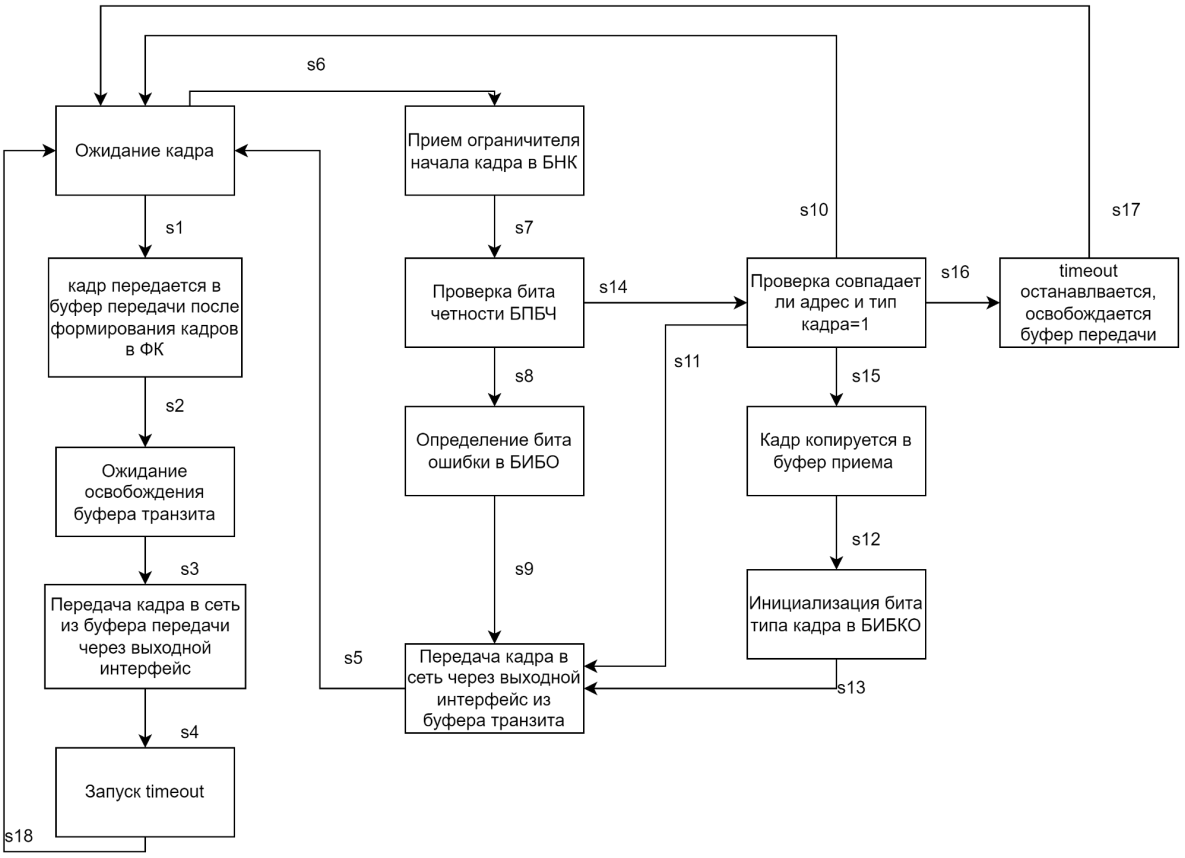

Рисунок 8 - Граф состояний сетевого контроллера

Рассматриваемые переходы:

1. s1 срабатывает, когда кадр поступает от станции
2. s2 срабатывает, когда данные и АП проходят через блок ФК, в котором задаются значения полям адреса получателя, отправителя, типа кадров, поля наличия ошибок, вычисляются биты четности
3. s3 срабатывает при освобождении буфера транзита
4. s4 срабатывает при передаче в сеть через выходной интерфейс
5. s6 срабатывает, когда кадр поступает через входной интерфейс
6. s7 срабатывает, когда поступает последовательность 01111110, которая означает, что следующие 7 бит являются данными адреса получателя
7. s14 срабатывает, если все поля Р совпадают с подсчитанными в этом блоке битами четности
8. s8 срабатывает, если в блоке БСПЧ обнаружена ошибка
9. s9 срабатывает срабатывает после инициализации поля БО, равной ошибке, обнаруженной в блоке проверки четности бита
10. s5 срабатывает при передаче кадра в сеть
11. s10 срабатывает, если адрес получателя и адрес отправителя не совпадают с текущей станцией, либо адрес отправителя совпадает с текущей станцией и ТК = 0.
12. s11 срабатывает, если адрес текущей станции не совпадает с АП и АО
13. s15 срабатывает, если адрес получателя совпадает с адресом текущей станции
14. s12 срабатывает, если кадр копируется в буфер приема
15. s13 срабатывает, если инициализирован тип кадра-ответа
16. s16 срабатывает, если адрес отправителя совпадает с адресом текущей станции и тип кадра = 1.

17. s17 срабатывает при очищении буфера, обнулении таймера и перевода станции в режим станции транзита или получателя
18. s18 срабатывает, если кадр передается в сеть

5. Разработка протокола подуровня MAC и описание алгоритма реализации процедуры передачи кадра в зависимости от заданного метода контроля

Формирование битов четности производится следующим образом. Для каждого поля кадра (за исключением поля флага начала, поля ТК и поля БО) высчитывается сумма значений битов по модулю два. Затем данный результат вносится в соответствующее поле Р.

При проверке данных происходит такая же операция – высчитывается сумма по модулю 2, а затем сравнивается со значением поля Р.

Если станция А хочет передать кадр станции В.

Формируется кадр. Устанавливается флаг начала кадра 01111110. Далее заносится адрес получателя и адрес отправителя. Тип кадра будет равен 0, так как кадр кадр данных. Обнаружение ошибок равно 0, потому что ошибок нет. В длину поля данных заносится количество байт, которые необходимо отправить. Сами данные помещаются в поле данных

Бит четности считается в каждом информационном поле, результат хранится в бите четности определенного поля. Запускается timeout. Соединение со средой передачи устанавливается. Когда каждая станция получает пакет данных, то определяет флаг начала. После получения кадра рабочая станция выполняет собственное вычисление по модулю 2, сравнивает полученный бит четности с принятым. Если бит четности не совпадает, то станция приемник формирует кадр-ответ с ошибкой. Сравнивается адрес получателя и отправителя с адресом текущей станции: если совпал адрес отправителя и при этом тип кадра 1, то кадр снимается с

сети, когда достигнет источника, timeout останавливается, а буфер передачи очищается. Если адрес совпал с адресом получателя и при этом тип кадра 0, то тип кадра меняется на 1, а кадр копируется в буфер приема. Если же адрес текущей станции не совпал с адресом отправителя или получателя, то кадр через буфер транзита идет дальше по сети.

Если в течение времени timeout станция А так и не получила кадр-ответ, то кадр посылается еще 10 раз. Кадр отбросится, если кадр-ответ не получен.

6. Граф-схема алгоритма процедуры передачи кадра в сети

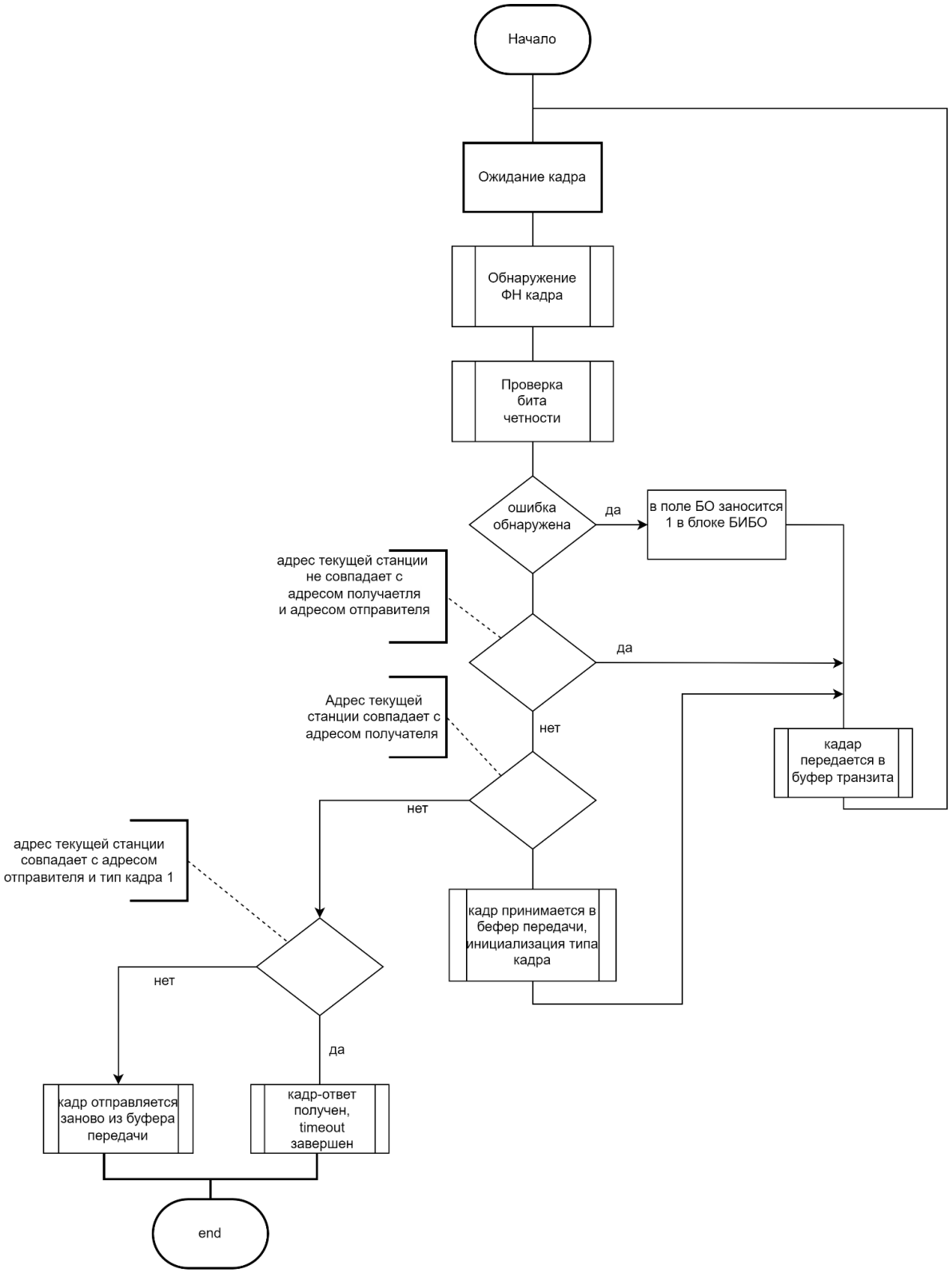

Рисунок 9 - Схема алгоритма для режима работы децентрализованного (транзитного) мониторного узла

7. Результаты моделирования

```
Введите адрес станции-отправителя:
10
Введите адрес станции-получателя:
20
Отправка кадра
Флаг начала:            01111110
Адрес отправителя:      0001010
Адрес получателя:       0010100
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            00110101011000100110000101001111000011010011100 1



Отправить кадр-ответ
1) Да
2) Нет
1
Флаг начала:            01111110
Адрес отправителя:      0010100
Адрес получателя:       0001010
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            00110101011000100110000101001111000011010011100 1
```

Рисунок 10 - Пример успешно отправленного кадра, бит ошибки = 0.

```
Введите адрес станции-отправителя:
20
Введите адрес станции-получателя:
33
Отправка кадра
Флаг начала:            01111110
Адрес отправителя:      0010100
Адрес получателя:       0100001
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            10100000010001100011000011000100000111011011100 1



Ошибка

Отправить кадр-ответ
1) Да
2) Нет
1
Флаг начала:            01111110
Адрес отправителя:      0100001
Адрес получателя:       0011100
Бит ошибки:             1
Длина поля данных:      000101
```

Рисунок 11 - Отправленный кадр с ошибкой в поле адрес отправителя, бит ошибки = 1.

```
Введите адрес станции-отправителя:
11
Введите адрес станции-получателя:
85
Отправка кадра
Флаг начала:            01111110
Адрес отправителя:      0001011
Адрес получателя:       1010101
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            1000111110101010001100101010111100101110111111000



Ошибка

Отправить кадр-ответ
1) Да
2) Нет
1
Флаг начала:            01111110
Адрес отправителя:      1010001
Адрес получателя:       0001011
Бит ошибки:             1
Длина поля данных:      000101
```

Рисунок 12 - Отправленный кадр с ошибкой в адресе получателя

```
Введите адрес станции-отправителя:
25
Введите адрес станции-получателя:
30
Отправка кадра
Флаг начала:            01111110
Адрес отправителя:      0011001
Адрес получателя:       0011110
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            10101111001011100100111011101110011000011010



Ошибка

Отправить кадр-ответ
1) Да
2) Нет
1
Флаг начала:            01111110
Адрес отправителя:      0011110
Адрес получателя:       0011001
Бит ошибки:             1
Длина поля данных:      000111
```

Рисунок 13 - Кадр с ошибкой в поле длина поля данных

```
Введите адрес станции-отправителя:
52
Введите адрес станции-получателя:
13
Отправка кадра
Флаг начала:            01111110
Адрес отправителя:      0110100
Адрес получателя:       0001101
Бит ошибки:             0
Длина поля данных:      000101
Поле данных:            11100010101001000001100001100101010010101100011



Ошибка

Отправить кадр-ответ
1) Да
2) Нет
1
Флаг начала:            01111110
Адрес отправителя:      0001101
Адрес получателя:       0110100
Бит ошибки:             1
Длина поля данных:      000101
Поле данных:            11100110101001000001100001100101010010101100011
```

Рисунок 14 - Кадр с ошибкой в поле данных

8. Анализ результатов

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.

Контроль по паритету представляет собой наиболее простой метод контроля данных. В то же время это наименее мощный алгоритм контроля, так как с его помощью можно обнаружить только одиночные ошибки в проверяемых данных.

Было выполнено моделирование работы станции в сети при заданных условиях и ограничениях. Результаты работы программы показывают, что программа работает корректно.

9. Приложение

9.1. main.cpp

```
#include "station.hpp"

int main() {
	setlocale(LC_ALL, "Russian");
	Station s;
}
```

9.2. station.hpp

```
#pragma once
#include "frame.hpp"
#include <time.h>
#include <stdlib.h>

class Station {
	short send[7];
	short recv[7];
	short control;
	short length[6];
	short data[48];
	Frame frame;
	Frame frameA;
	int N = 90;

public:
	Station() {
		int ch;
```

```
            cout << "1) Отправка кадра" << endl;
            cout << "2) Ошибка в адресе отправителя" << endl;
            cout << "3) Ошибка в адресе получателя" << endl;
            cout << "5) Ошибка в длине поля данных" << endl;
            cout << "6) Ошибка в данных" << endl;
            cin >> ch;
            cout << endl;
            system("cls");
            initP();
            frame.init(send, recv, control, length, data);
            cout << "Отправка кадра" << endl;
            frame.show();

            switch (ch)
            {
            case 1:
                 break;
            case 2:
                 frame.send[3] = (frame.send[3] == 1) ? 0 : 1;
                 break;
            case 3:
                 frame.recv[4] = (frame.recv[4] == 1) ? 0 : 1;
            case 4:
                 frame.control = (frame.control == 1) ? 0 : 1;
                 break;
            case 5:
                 frame.length[4] = (frame.length[4] == 1) ? 0
: 1;
                 break;
            case 6:
                 frame.data[5] = (frame.data[5] == 1) ? 0 : 1;
                 break;
            default:
                 exit(1);
                 break;
            }
```

```
            int j = 0;
            for (int i = 0; i < 7; i++)
                j += frame.recv[i] * pow((float)2, 6 - i);
            cout << endl;

            j++;
            j = j % N;
            cout << endl;

            if (frame.check() == true) {
                cout << "Адрес отправителя == Адрес
получателя. Пакет отброшен" << endl;
                return;
            }

            int num = 16;
            while (num != 0) {
                cout << endl;
                int res = par(j);
                if (res == 0) {
                    cout << endl;
                    cout << "Отправить кадр-ответ" << endl;
                    cout << "1) Да" << endl;
                    cout << "2) Нет" << endl;
                    cin >> res;
                    if (res == 1) {
                        frameA.show();
                        return;
                    }
                    else {
                        cout << "Адрес не может быть
достигнут" << endl;
                        cout << "Кадр будет отправлен еще "
<< num - 1 << "  раз" << endl;
                        num--;
```

```
                    }
                }
            }
        }

        void initP() {
            send[0] = 0;
            send[1] = 0;
            send[2] = 0;
            send[3] = 0;
            send[4] = 0;
            send[5] = 0;
            send[6] = 0;

            recv[0] = 0;
            recv[1] = 0;
            recv[2] = 0;
            recv[3] = 0;
            recv[4] = 0;
            recv[5] = 0;
            recv[6] = 0;

            int s, r;
            int buf[7];
            int k = 1;
            cout << "Введите адрес станции-отправителя:" <<
endl;
            cin >> s;
            cout << "Введите адрес станции-получателя:" <<
endl;
            cin >> r;
            int i = 0;
            while (s) {
                send[6 - i] = (s % 2);
                k *= 10;
                s /= 2;
```

```
            i++;
        }
        k = 1;
        i = 0;
        while (r) {
            recv[6 - i] = (r % 2);
            k *= 10;
            r /= 2;
            i++;
        }

        control = 0;

        length[0] = 0;
        length[1] = 0;
        length[2] = 0;
        length[3] = 1;
        length[4] = 0;
        length[5] = 1;

        srand(time(NULL));

        for (int i = 0; i < 48; i++)
            data[i] = (rand()) % 2;
}

int receive(Frame frame, int j) {
        int counter;
        int station[7];
        int stationRecv[7];
        counter = 0;
        int buf = 0;
        int k = j;
        int flag = 0;
        short parity_send = 0;
        short parity_recv = 0;
```

```
short parity_control = 0;
short parity_length = 0;
short parity_data[48];

for (int i = 0; i < 7; i++) {
     station[6 - i] = k % 2;
     k = (j - station[6 - i]) / 2;
     j = k;
}

for (int i = 0; i < 7; i++) {
     parity_send += frame.send[i];
}
parity_send %= 2;

for (int i = 0; i < 7; i++) {
     parity_recv += frame.recv[i];
}
parity_recv %= 2;
parity_control += frame.control;
parity_control %= 2;

for (int i = 0; i < 6; i++) {
     parity_length += frame.length[i];
}
parity_length %= 2;

for (int i = 0; i < 48; i++) {
     parity_data[i] += frame.data[i] % 2;
}

if (frame.parity_send != parity_send)
     return 1;

if (frame.parity_recv != parity_recv)
     return 1;
```

```
    if (frame.parity_control != parity_control)
        return 1;

    if (frame.parity_length != parity_length)
        return 1;

    for (int i = 0; i < 48; i++)
        if (frame.parity_data[i] != parity_data[i])
            return 1;

    for (int i = 0; i < 7; i++)
        if (frame.recv[i] == station[i])
            counter++;

    if (counter != 7)
        return 0;

    for (int i = 0; i < 7; i++)
        if (frame.send[i] == station[i])
            counter++;

    if (counter == 7)
        return 6;

    for (int i = 0; i < 7; i++)
        buf += frame.send[i] * pow((float)2, 7 - i);

    if (buf > N)
        return 11;

    return 10;
}

int par(int j) {
    int res = 2;
```

```
        for (int i = 0; i < N; i++) {
            j = j % N;
            int fl = receive(frame, j);
            switch (fl)
            {
            case 1:
                control = 1;
                cout << "Ошибка" << endl;
                frameA.initA(frame.send, frame.recv,
control, frame.length, frame.data);
                return 0;
                break;

            case 6:
                cout << "Кадр-ответ успешно доставлен"
<< endl;
                return 1;
                break;

            case 10:
                frameA.initA(frame.send, frame.recv,
control, frame.length, frame.data);
                return 0;
                break;
            case 11:
                cout << "Адрес доставки больше 90" <<
endl;
                return 0;
                break;
            default:
                break;
            }
            j++;
        }
        cout << "Кадр успешно вернулся к источнику и был
отброшен" << endl;
```

```
        return res;
    }
};
```

9.3. frame.hpp

```
#pragma once
#include <iostream>

using namespace std;

class Frame {
public:
    short start[8];
    short send[7];
    short parity_send = 0;
    short recv[7];
    short parity_recv = 0;
    short control = 0;
    short parity_control = 0;
    short length[6];
    short parity_length = 0;
    short data[48];
    short parity_data[48];

    void init(short send[7], short recv[7], short control,
short length[6], short data[48]) {
        this->start[0] = 0;
        this->start[1] = 1;
        this->start[2] = 1;
        this->start[3] = 1;
        this->start[4] = 1;
        this->start[5] = 1;
        this->start[6] = 1;
        this->start[7] = 0;
```

```
        for (int i = 0; i < 7; i++) {
            this->send[i] = send[i];
            parity_send += send[i];
        }
        parity_send %= 2;

        for (int i = 0; i < 7; i++) {
            this->recv[i] = recv[i];
            parity_recv += recv[i];
        }
        parity_recv %= 2;

        this->control = control;
        parity_control += control;

        parity_control %= 2;

        for (int i = 0; i < 6; i++) {
            this->length[i] = length[i];
            parity_length += length[i];
        }
        parity_length %= 2;

        for (int i = 0; i < 48; i++) {
            this->data[i] = data[i];
            parity_data[i] += data[i] % 2;
        }
    }

    void show() {
        cout << "Флаг начала: \t\t";
        for (int i = 0; i < 8; i++)
            cout << start[i];
        cout << endl;

        cout << "Адрес отправителя: \t";
```

```
        for (int i = 0; i < 7; i++)
            cout << send[i];
        cout << endl;

        cout << "Адрес получателя: \t";
        for (int i = 0; i < 7; i++)
            cout << recv[i];
        cout << endl;

        cout << "Бит ошибки: \t\t";
        cout << control;
        cout << endl;

        cout << "Длина поля данных: \t";
        for (int i = 0; i < 6; i++)
            cout << length[i];
        cout << endl;

            cout << "Поле данных: \t\t";
            for (int i = 0; i < 48; i++)
                cout << data[i];
        cout << endl;
    }

    void initA(short send[7], short recv[7], short
control, short length[6], short data[48]) {
        this->start[0] = 0;
        this->start[1] = 1;
        this->start[2] = 1;
        this->start[3] = 1;
        this->start[4] = 1;
        this->start[5] = 1;
        this->start[6] = 1;
        this->start[7] = 0;

        for (int i = 0; i < 7; i++) {
```

```
            this->send[i] = recv[i];
            parity_send += recv[i];
        }
        parity_send %= 2;

        for (int i = 0; i < 7; i++) {
            this->recv[i] = send[i];
            parity_recv += send[i];
        }
        parity_recv %= 2;

        this->control = control;
        parity_control += control;

        parity_control %= 2;

        for (int i = 0; i < 6; i++) {
            this->length[i] = length[i];
            parity_length += length[i];
        }
        parity_length %= 2;

        for (int i = 0; i < 48; i++) {
            this->data[i] = data[i];
            parity_data[i] += data[i] % 2;
        }
    }

    bool check() {
        for (int i = 0; i < 7; i++) {
            if (send[i] != recv[i])
                return false;
        }
        return true;
    }
};
```