Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«___» _____2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Моделирование и прототипирование протокола канального уровня спутниковой системы
тема

Руководитель	_____	старший преподаватель	И.Н.Рыженко
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		А.А.Коряпин
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	старший преподаватель	И.Н.Рыженко
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«___» _____2023 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Коряпину Андрею Алексеевичу
фамилия, имя, отчество

Группа КИ19-06Б Направление (специальность) 090301
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы: Моделирование и прототипирование протокола канального уровня спутниковой системы

Утверждена приказом по университету № 4765/С от

Руководитель ВКР: И.Н. Рыженко, старший преподаватель кафедры ВТ
инициалы, фамилия, учёная степень, должность, место работы

Исходные данные для ВКР:

1) Разработать протокол канального уровня спутниковой системы на базе Стандарта ETSI EN 301 545-2 V1.2.1 (2014-04)

2) Входной пакет инкапсулятора IP (RFC 791)

3) Протокол должен обеспечивать возможность передачи адреса источника/передатчика

4) Предусмотреть алгоритм обнаружения ошибок, возникающих при Искажении передаваемой информации по каналу связи

5) Размер полезной нагрузки пакета канального уровня от 10 до 1500 байт

Перечень разделов ВКР:

1) Анализ предметной области

2) Разработка модели

3) Реализация модели

Перечень графического материала: Презентация доклада выступления

Руководитель ВКР _____ И.Н. Рыженко
подпись инициалы, фамилия

Задание принял к исполнению _____ А.А. Коряпин
подпись инициалы, фамилия

« » _____ _____ г.
дата

РЕФЕРАТ

Настоящая бакалаврская работа посвящена разработке прототипа и моделированию протокола канального уровня спутниковой системы.

Данная пояснительная записка содержит 70 страниц текста с иллюстрациями, 7 приложений и 5 использованных источников.

RETURN LINK ENCAPSULATION, ПРОТОКОЛЫ ИНКАПСУЛЯЦИИ, ИНКАПСУЛЯЦИЯ ДАННЫХ, СПУТНИКОВЫЕ СИСТЕМЫ.

Цель бакалаврской работы – построение прототипа и моделирование протокола канального уровня спутниковой системы.

Задачи, решённые в процессе разработки:

– разработана архитектура модели;

– разработаны составные модули модели;

– проведено тестирование разработанной модели;

– исследован протокол инкапсуляции обратного канала.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В наше время все большее значение приобретают информационные технологии, мобильная связь, телевидение. Ведущую роль играют беспроводная и проводная сеть Интернет, которая является не только источником информации, но и универсальной средой для общения, развлечения и обучения. В жизни современного человека он представляет собой необходимый инструмент, помогающий в работе, общении и отдыхе.

В больших городах любой человек может себе позволить высокоскоростной доступ в Интернет. Но в мире всё ещё существуют места, где доступ к нему затруднён по различным причинам. Например, в отдалённые малонаселённых пунктах, куда прокладывать оптоволоконный кабель может быть очень затратно и не выгодно. С беспроводными LTE станциями может возникнуть аналогичная проблема. В некоторых случаях подобные технологии использовать просто не получится, например, при обеспечении морских судов связью и интернетом. Поэтому спутниковая связь до сих пор является актуальной и востребованной.

По этой причине наше студенческое конструкторское бюро разрабатывает свою собственную спутниковую сеть коротких сообщений, ориентированную для использования с интернетом вещей. Трафик в такой сети будет распространяться в TCP/IP пакетах. По этой причине возникла необходимость в использовании спутниковых протоколов связи для поддержки данного вида трафика. Одним из таких протоколов является протокол инкапсуляции обратного канала, способный адаптировать TCP/IP трафик под спутниковую связь на канальном уровне. В ходе этой работы будет построена его программная модель.

Целью работы является построение модели протокола канального уровня спутниковой системы. Необходимо определить основные модули, их задачи и

используемые в них форматы данных основываясь на существующих стандартах для данной отрасли.

Задачи работы:

1. Проанализировать существующие стандарты для данной отрасли;
2. Разработка архитектуры программной модели;
3. Реализовать основные модули для разрабатываемой модели;
4. Протестировать разработанные модули.

1 Анализ предметной области

1.1 Анализ задания

Необходимо построить модель протокола инкапсуляции канального уровня для спутниковой системы. Главной задачей модели является адаптация пакетов сетевого уровня (например IP) к их передаче по спутниковому каналу связи посредством инкапсуляции в собственные форматы данных.

Модель будет состоять из двух частей - передатчика и приёмника.

Задачи передатчика:

– принять на входе пакеты сетевого уровня модели OSI с дополнительной информацией о них и пакеты внутренней сигнализации спутниковой сети;

– инкапсулировать пришедшие пакеты в свои форматы данных;

– при необходимости разделить инкапсулированные пакеты на фрагменты (в зависимости от размера пакета и будущего кадра);

– отсортировать целые или фрагментированные пакеты в порядке их приоритета (на основе QoS-тэга, привязанного к каждому из них);

– поместить один или несколько пакетов в кадр для последующей передачи по физическому каналу.

Задачи приёмника:

– принять кадры из физического канала, извлечь из них вложенные пакеты;

– отфильтровать полученные пакеты по их типу (целые, фрагментированные, сигнальные);

– собрать фрагментированные пакеты, проверив их целостность;

– изъять из пакетов собственного формата исходные пакеты сетевого уровня или внутренней сигнализации;

1.1.1 Распределение задач по уровням

Задачи протокола можно разбить на 3 уровня:

1) Уровень инкапсуляции: на данном этапе пакеты сетевого уровня инкапсулируются в собственный формат данных и извлекаются из него. На рисунке 1 изображён в виде блока «Encapsulator» передатчика и «Decapsulator» приёмника.

2) Уровень фрагментации: на данном этапе инкапсулированные пакеты разделяются на части и собираются в исходный инкапсулированный пакет. На рисунке 1 изображён в виде блока «Fragmenter» передатчика и «Reassembler» приёмника.

3) Уровень формирования сообщения: на данном этапе из фрагментированных пакетов формируется сообщение, передающееся на физический уровень. Далее это сообщение принимается и из него извлекаются все фрагменты. На рисунке изображён в виде блока «Burst packing» передатчика и «Burst unpacking» приёмника.

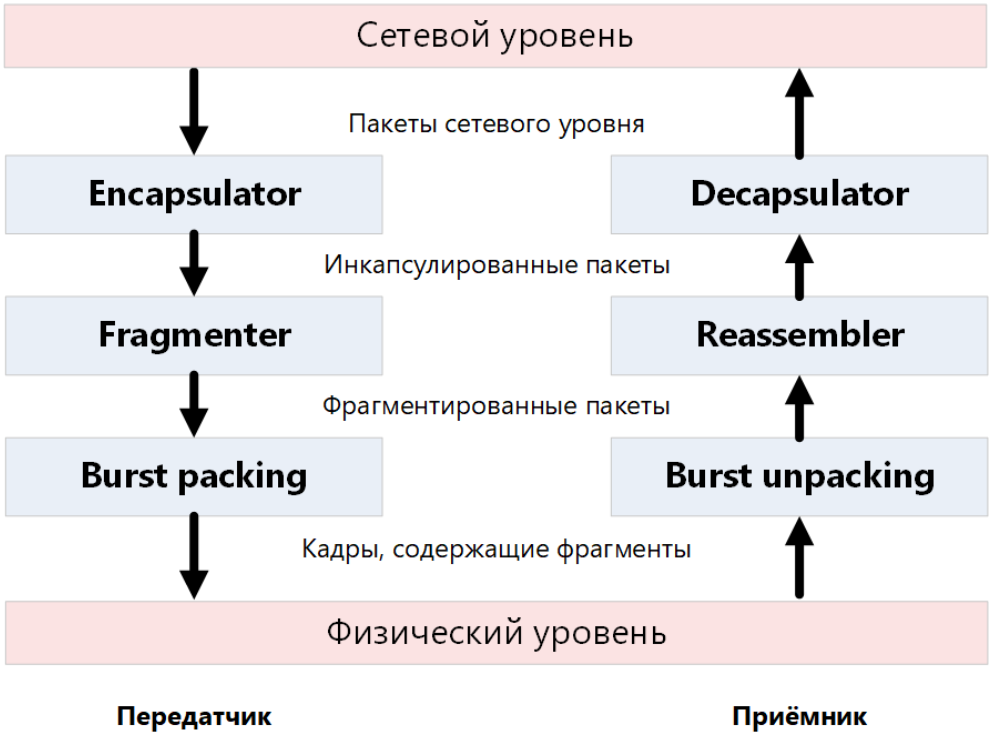


Рисунок 1 — Уровни разбиения задач

1.1.2 Форматы данных модели и их связь между собой

Протокол будет использовать следующие форматы данных:

– Addressed Link PDU (ALPDU): данный формат будет необходим для того, чтобы модули сетевого уровня приёмника могли определить обработчик для вложенного пакета.

– Payload-adapted PDU (PPDU): данный формат будет необходим для того, чтобы входные пакеты можно было передавать по частям и, в последствии, собрать эти части в исходный пакет.

– Frame PDU: данный формат будет необходим для адресации в спутниковой сети и возможности передать несколько пакетов одним сообщением.

Взаимосвязь форматов данных показана на рисунке 2.

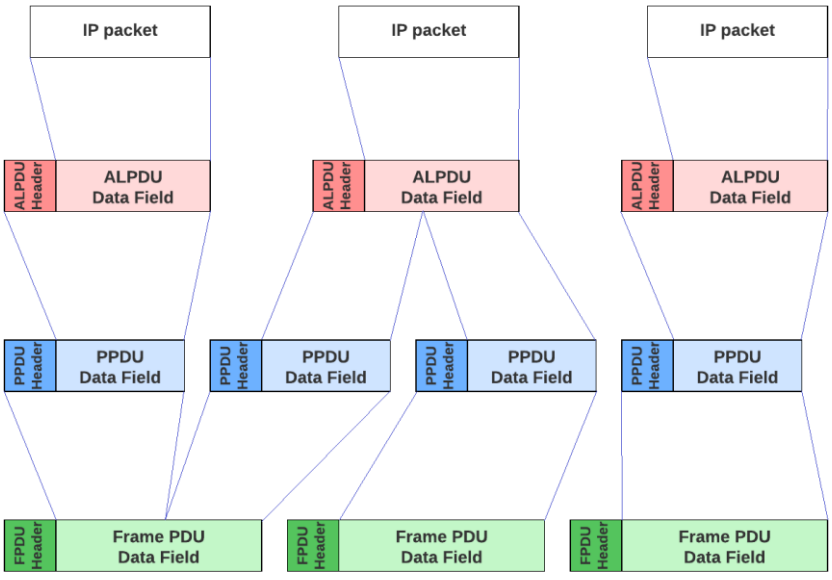


Рисунок 2 — Взаимосвязь форматов данных

Исходный IP пакет сначала инкапсулируется в ALPDU пакет. Далее, в зависимости от размера IP пакета, ALPDU пакет либо полностью помещается в

PPDU пакет, либо фрагментируется и помещается в несколько PPDU пакетов. Дальше происходит процесс инкапсуляции пакетов PPDU в Frame PDU. В Frame PDU помещается либо один, либо несколько PPDU пакетов.

Процесс деинкапсуляции исходного пакета происходит в обратном порядке. В зависимости от параметров указанных в заголовках, пакеты верхнего уровня деинкапсулируются и дефрагментируются, образуя исходный IP пакет.

1.1.3 Структурная схема разрабатываемой модели

Исходя из анализа структур разрабатываемого протокола и их связи между собой можно построить упрощённую структурную схему для разрабатываемой модели (Рисунок 3).

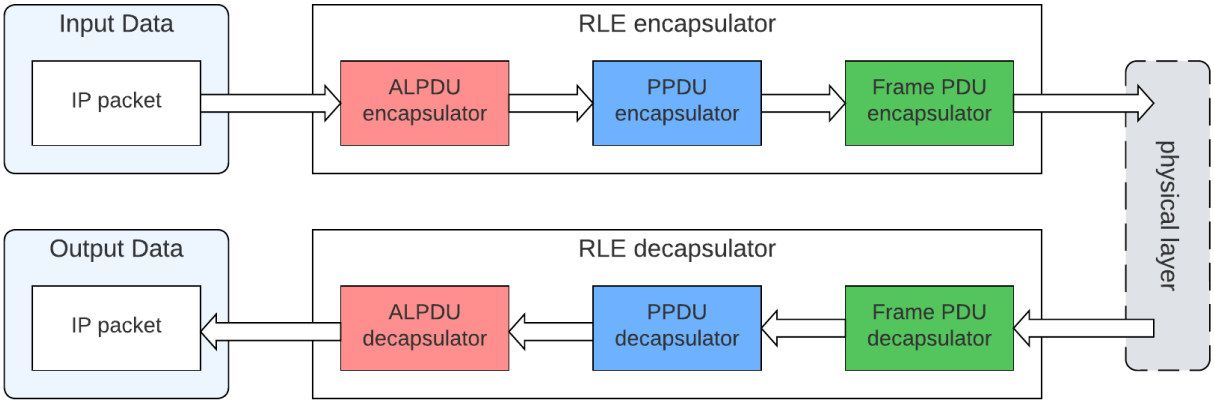


Рисунок 3 — Структурная схема модели

Основываясь на построенной схеме, можно однозначно разделить разрабатываемую модель на 2 части, в каждом из которых будет по 3 модуля. Разработку каждого модуля можно выделить как отдельный этап разработки общей модели.

1.2 Функциональные требования к модели

Требования к модели:

– входной пакет инкапсулятора: IP (RFC 791);

– размер полезной нагрузки пакета канального уровня: от 10 до 1500 байт;

– предусмотреть алгоритм обнаружения ошибок, возникающих при искажении передаваемой информации по каналу связи;

– должна быть возможность фрагментации исходного пакета на несколько частей при инкапсуляции;

– в модель должна быть включена дополнительная структура, отвечающая за конфигурационные параметры центральной станции, влияющие на некоторые опциональные поля в ранее описанных форматах данных.

1.3 Инструменты разработки

Модель будет представлена в виде ПО на языке C++, для разработки будет применяться интегрированная среда разработки Microsoft Visual Studio Community 2022.

1.4 Анализ существующих решений

На мировом рынке представлено несколько компаний, которые имеют реализацию данного протокола согласно стандарту DVB-RCS2 - HughesNet, Gilat Satellite Networks, IDirect, ViaSat и т.д. Каждая из них имеет свою реализацию разрабатываемого протокола, так как стандарт оставляет свободу для своих решений, но в открытый доступ её не выкладывают по коммерческим причинам. Из-за этого анализ существующих решений провести сложно, известно только то, что они уже введены в эксплуатацию в действующих спутниковых системах.

1.5 Актуальность работы

Актуальность обусловлена задачей передачи данных в разрабатываемой системе спутниковой связи и отсутствием открытых существующих решений.

1.6 Выводы по первой главе

В ходе написания главы было проанализировано полученное задание, существующие решения, была сформулирована цель работы и разработана концептуальная структура для разрабатываемой модели.

2 Разработка модели

Ранее было определено, что модель будет состоять из 2 частей - передатчика и приёмника.

Структурная схема передатчика показана на рисунке 4.

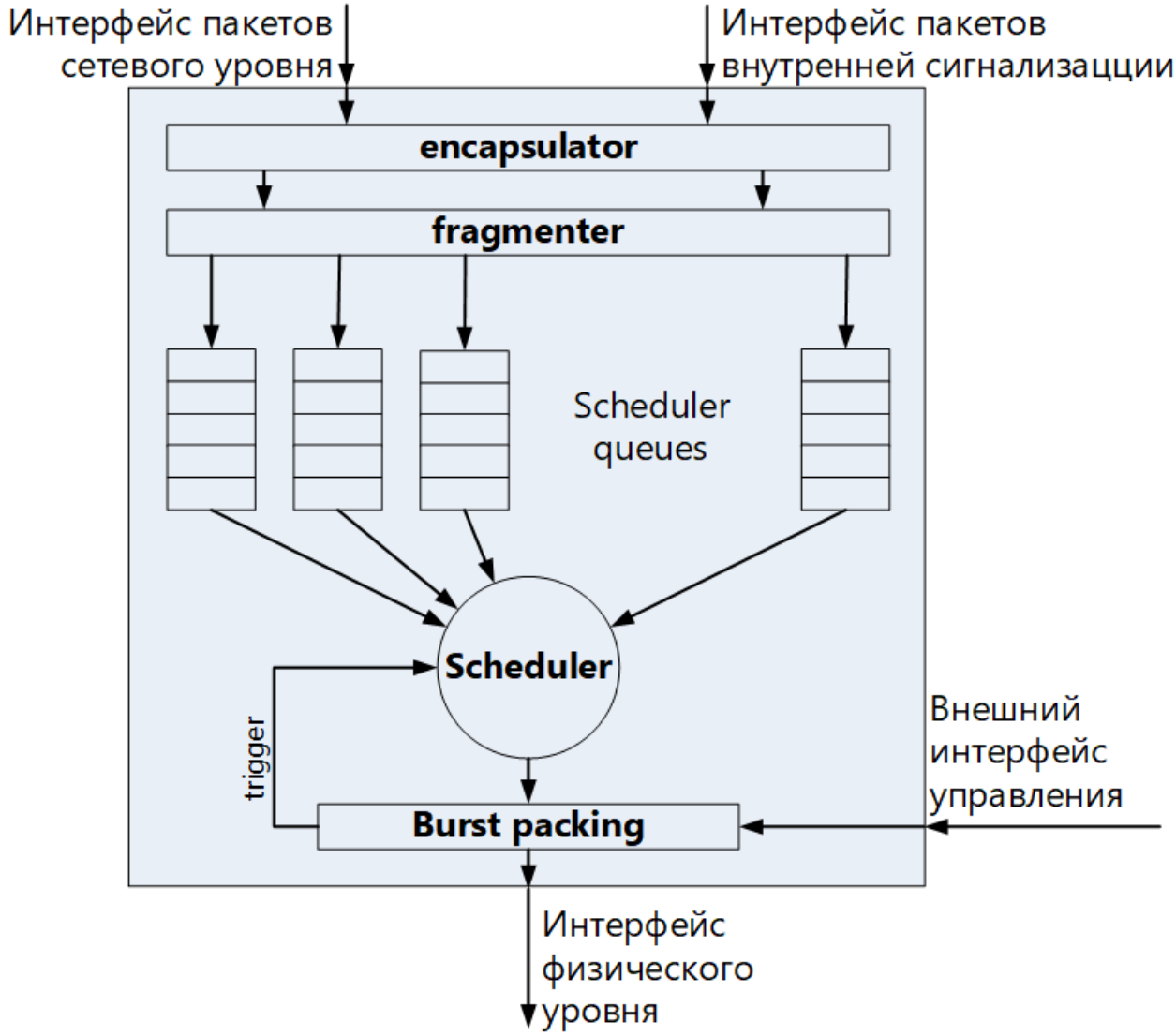


Рисунок 4 — Структурная схема передатчика

Модуль «Encapsulator» принимает входные пакеты с внешних интерфейсов и формирует ALPDU пакеты. Модуль «Fragmenter» принимает ALPDU пакеты от модуля «Encapsulator», при необходимости их фрагментирует, формирует PPDU пакеты и помещает их в очереди, исходя из

их приоритета. Модуль «Burst packing» по активации с внешнего интерфейса запрашивает PPDU пакеты у модуля Scheduler и образует кадры Frame PDU, которые далее отправляет на физический уровень.

Описание внешних интерфейсов:

– интерфейс пакетов сетевого уровня генерирует структуры, содержащие пакеты сетевого уровня;

– интерфейс пакетов внутренней сигнализации генерирует структуры, содержащие пакеты внутренней сигнализации спутниковой сети;

– внешний интерфейс управления активирует модуль «Burst packing», а также передаёт ему основные параметры;

– интерфейс физического уровня принимает сформированные кадры Frame PDU и передаёт их на физический уровень;

Структурная схема приёмника показана на рисунке 5.

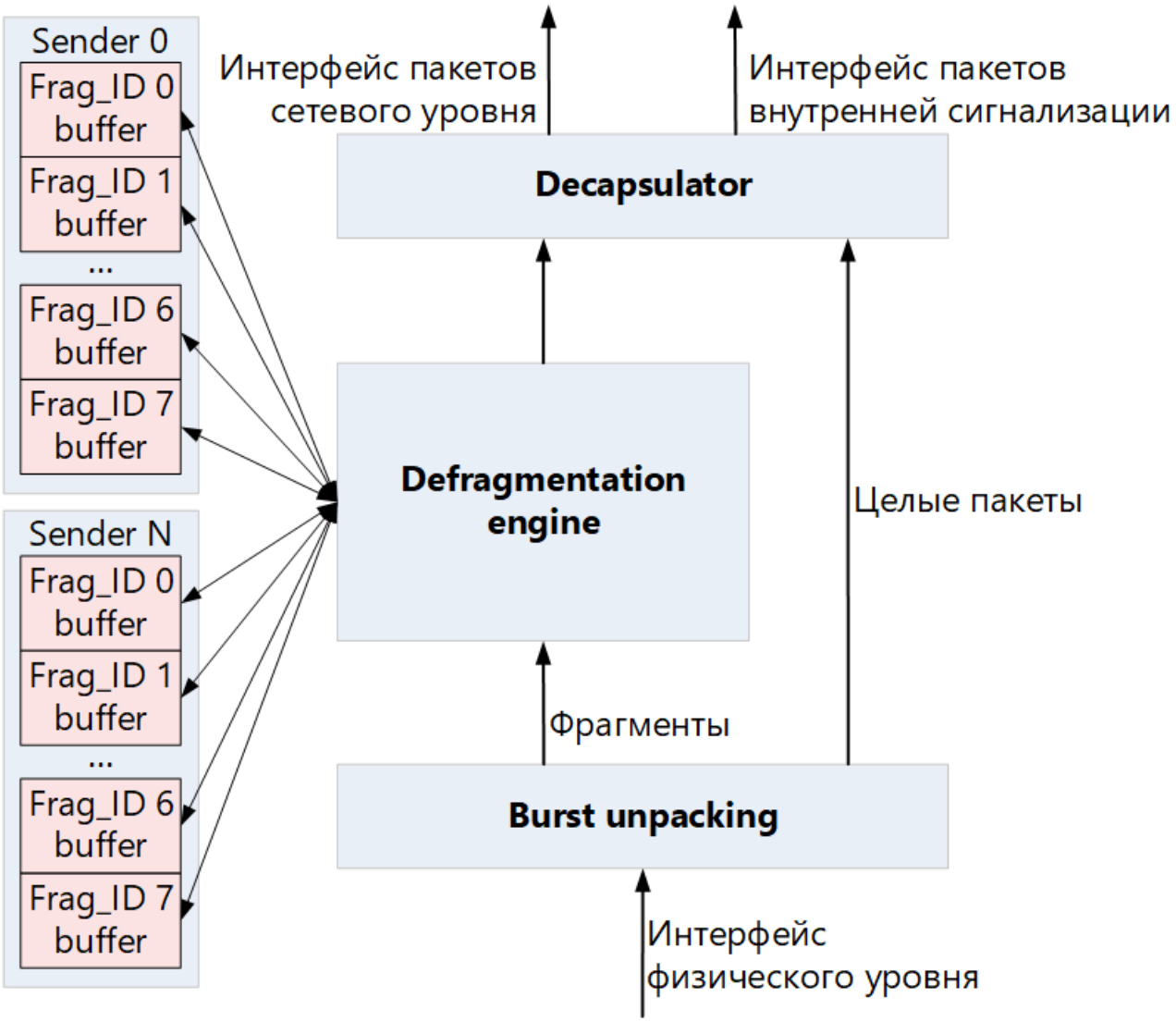


Рисунок 5 — Структурная схема приёмника

Модуль «Burst unpacking» принимает с физического уровня кадры Frame PDU, извлекает из них PPDU пакеты и отправляет их в модули «Defragmentation engine» и «Decapsulator», в зависимости от их заголовка. «Defragmentation engine» осуществляет сборку поступивших PPDU пакетов и извлекает из них ALPDU пакет, который далее отправляется в модуль «Decapsulator». При этом данный модуль имеет в себе контейнеры, хранящие PPDU пакеты от разных отправителей. В свою очередь, каждый контейнер может хранит в себе PPDU пакеты, принадлежащие различным ALPDU пакетам (до 8 штук). Модуль «Decapsulator» извлекает из ALPDU пакета исходный пакет.

Описание внешних интерфейсов:

– интерфейс пакетов сетевого уровня принимает структуры, содержащие пакеты сетевого уровня;

– интерфейс пакетов внутренней сигнализации принимает структуры, содержащие пакеты внутренней сигнализации спутниковой сети;

– интерфейс физического уровня передаёт пришедшие кадры Frame PDU и передаёт их модулю «Burst unpacking»;

2.1 Структуры параметров модели

Общая структура параметров для передатчика и приёмника представлена на рисунке 6.

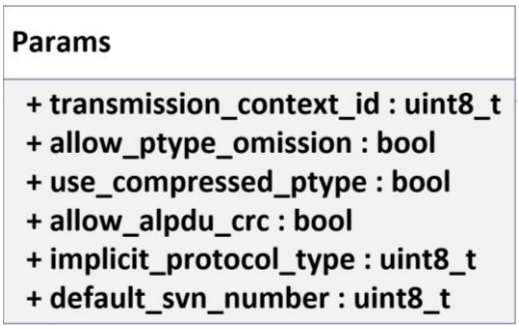


Рисунок 6 — Общая структура параметров

Описание полей:

• поле transmission_context_id содержит id контекста передачи. Данное поле влияет на заголовок Frame PDU пакета. Всевозможные значения указаны в таблице 1;

• поле use_compressed_ptype содержит в себе флаг, указывающий нужно ли использовать укороченный вариант EtherType записи типа протокола (compressed_protocol_type) в заголовке ALPDU пакета. Всевозможные значения compressed_protocol_type указаны в таблице 2;

- поле implicit_protocol_type хранит в себе одно из значений compressed_protocol_type и указывает на стандартный тип(-ы) протокола, используемый(-ые) при передаче данных. Влияет на заголовок ALPDU пакета;
- поле allow_ptype_omission содержит в себе флаг, указывающий на исключение типа протокола из заголовка ALPDU пакета, если он соответствует значению ранее указанного поля implicit_protocol_type;
- поле allow_alpdu_crc содержит в себе флаг, указывающий на тип алгоритма защиты, используемого при фрагментации входных пакетов. Влияет на заголовок ALPDU пакета и алгоритм работы модуля «Fragmenter»;
- поле default_svn_number хранит в себе старший байт адреса стандартной спутниковой виртуальной сети, в которой происходит передача данных. Влияет на заголовок ALPDU пакета.

Таблица 1 – Возможные значения поля transmission_context_id

transmission_context_id	**Контекст передачи**
0	Transparent star, dedicated access
1	Transparent star, Slotted Aloha access
2	Transparent star, CRDSA access

Таблица 2 – Возможные значения поля compressed_protocol_type

Compressed protocol type value	**Protocol ethertype value**	**Protocol**
0x0D	0x0800	IPv4
0x0E	0x0806	ARP
0x0F	0x8100	VLAN tagged frame
0x10	0x22F1	ROHC
0x11	0x86DD	IPv6
0x12	0x8809	Slow Protocols (IEEE 802.3)
0x13	0x8847	MPLS unicast
0x14	0x8848	MPLS multicast
0x15	0x8863	PPPoE Discovery Stage
0x16	0x8864	PPPoE Session Stage
0x17	0x888E	EAP over LAN
0x18	0x8906	Fibre Channel over Ethernet
0x19	0x88A8	Q-in-Q (IEEE 802.1ad)
0x1A	0x9100	Q-in-Q

Окончание таблицы 2

Compressed protocol type value	Protocol ethertype value	Protocol
0x30	0x0800, 0x86DD	IPv4, IPv6
0x42	0x0082	Internal M&C signalling (L2S)
0xFF		Для неподдерживаемых протоколов

Дополнительная структура параметров передатчика представлена на рисунке 7. Поля данной структуры могут быть вообще не задействованы и используются в зависимости от контекста передачи. Применяются для формирования заголовка Frame PDU.

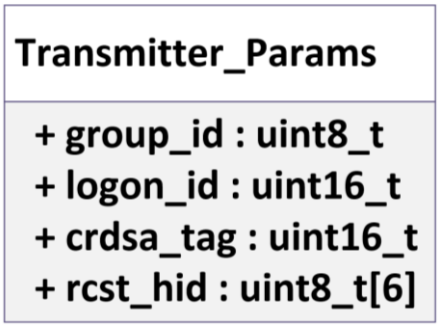


Рисунок 7 — Дополнительная структура параметров передатчика

Описание полей:

- поля group_id и logon_id хранят значения идентификации, полученные передатчиком при авторизации;
- поле rcst_hid хранит 6 байт уникального номера передатчика;
- поле crdsa_tag хранит 2-байтный тэг, необходимый для передачи данных при использовании контекста передачи CRDSA.

2.2 Разработка модулей передатчика

2.2.1 Разработка модуля «Encapsulator»

Входные данные:

– пакет Service Data Unit (SDU): входной пакет сетевого уровня (например IPv4) или внутренней сигнализации спутниковой сети;

– значение protocol_type: тип протокола входного пакета (для пакетов внутренней сигнализации фиксирован значением 0x0082);

– значение SVN_Source: старший байт адреса спутниковой виртуальной сети, из которой был принят входной пакет (для пакетов внутренней сигнализации может отсутствовать);

– значение QoS_tag: тэг приоритета, с которым был принят пакет (для пакетов внутренней сигнализации может отсутствовать);

Если какой-либо параметр отсутствует, то он принимает стандартное значение исходя из общей структуры параметров или наивысшее значение (в случае QoS-тэга).

Задачи модуля:

– сформировать ALPDU заголовок для входного пакета;

– инкапсулировать входной пакет в ALPDU пакет;

– передать сформированный ALPDU пакет с дополнительными параметрами модулю «Fragmenter».

Формируемый пакет данных: Addressed Link PDU.

Его структура представлена на рисунке 8.

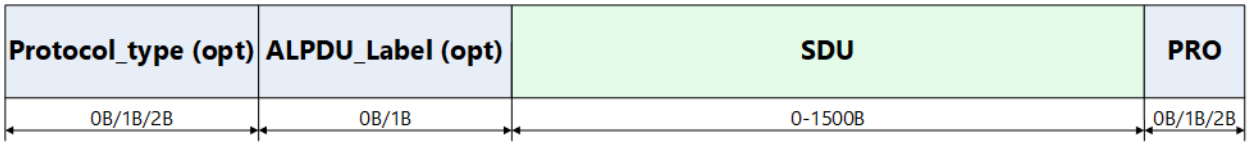


Рисунок 8 — Структура формата ALPDU

Описание полей данного формата:

– поле Protocol-Type хранит значение формата EtherType, принадлежащее протоколу пакета, хранимого в поле SDU. Это поле опционально, хранимое значение и размер зависят от полей allow_ptype_omission и use_compressed_ptype общей структуры параметров, описанных ранее. Длина: 0-2 байта;

– поле ALPDU_Label хранит в себе метку, прикрепленную к пакету более высокого уровня (MAC-адрес или номер SVN). Это поле опционально и может отсутствовать в случае, если входное значение protocol_type соответствует значению implicit_protocol_type из общей структуры параметров, описанной ранее. Длина: 0-1 байт(-ов);

– поле SDU хранит в себе непосредственно входной пакет сетевого уровня (например пакет IPv4) или внутренней сигнализации спутниковой сети. Длина: 0-1500 байтов;

– поле PRO хранит в себе контрольную сумму алгоритма CRC или Sequence_number, в зависимости от используемого алгоритма защиты, задаваемым исходя из поля use_alpdu_crc общей структуры параметров. Применяется только в случае фрагментации на части данного формата данных, по этой причине решение об его включении принимается на этапе обработки модулем «Fragmenter». Длина: 0-4 байта.

Помимо ALPDU пакета, данный модуль формирует дополнительные параметры, передаваемые модулю «Fragmenter».

Описание дополнительных параметров:

– ALPDU_Label_type: принимает различные значения в зависимости от входных значений protocol_type и SVN_Source. Алгоритм вычисления ALPDU_Label_type представлен на рисунке 9.

– protocol_type_suppressed: является флагом и принимает значение единицы, если в ALPDU пакете поле Protocol_type отсутствует.

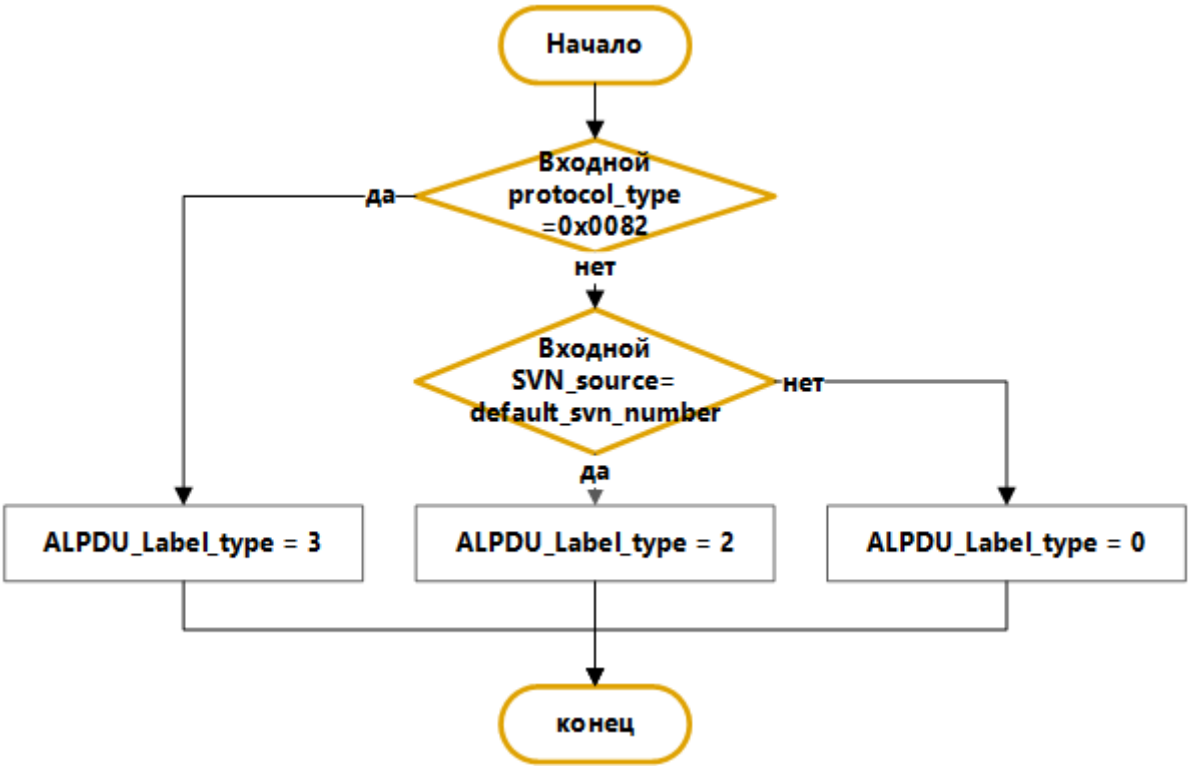


Рисунок 9 — Алгоритм вычисления ALPDU_Label_type

Выходные данные:

– сформированный ALPDU пакет;

– вычисленное значение ALPDU_Label_type;

– вычисленное значение protocol_type_suppressed;

– значение QoS_tag.

Алгоритм работы данного модуля представлен на рисунке 10.

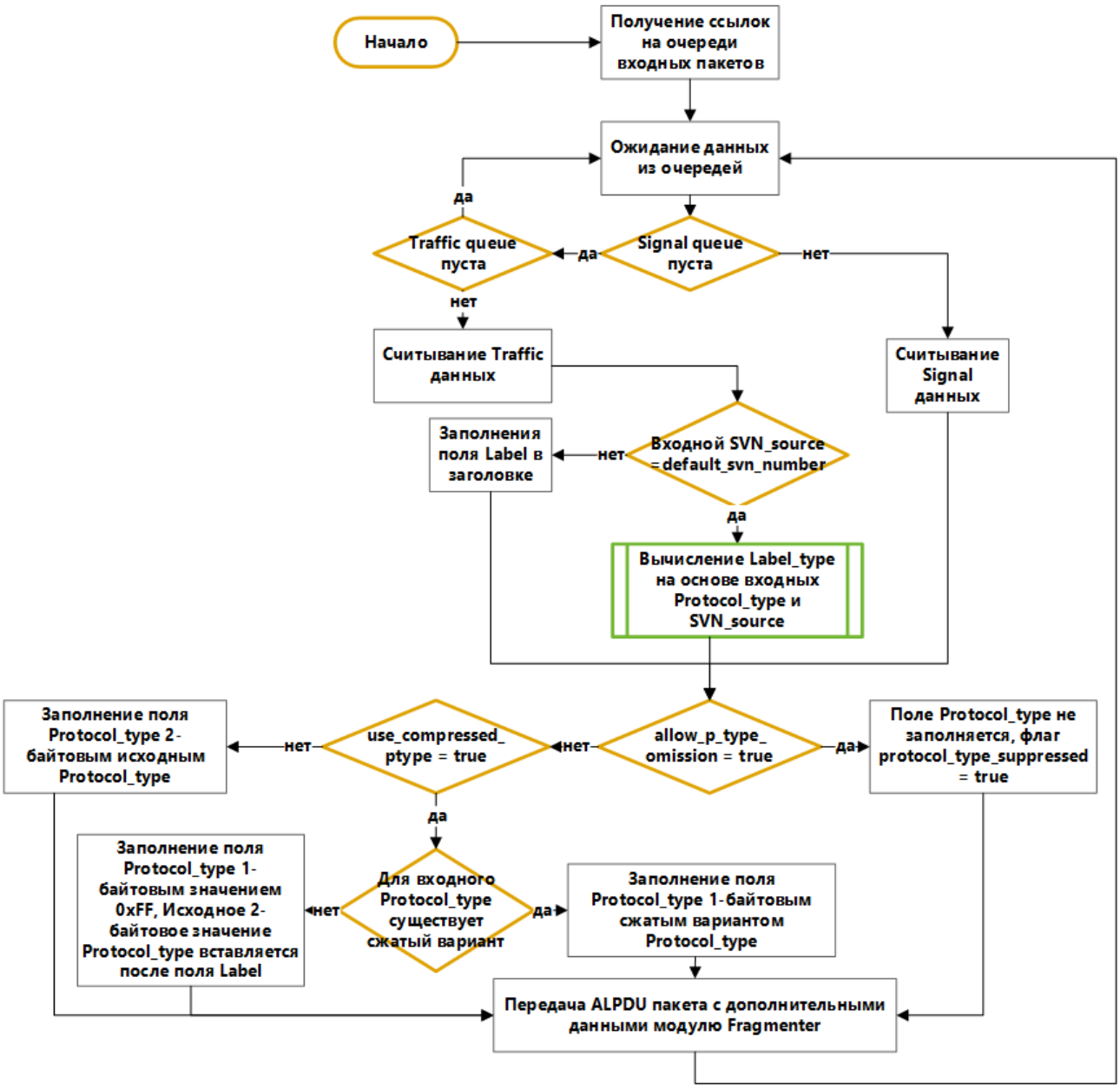


Рисунок 10 — Алгоритм работы модуля «Encapsulator»

2.2.2 Разработка модуля «Fragmenter»

Входные данные:

– пакет ALPDU: пакет, сформированный модулем «Encapsulator»;

– значение ALPDU_Label_type: значение, вычисленное модулем «Encapsulator»;

– значение protocol_type_suppressed: значение, вычисленное модулем «Encapsulator»;

– значение QoS_tag: тэг приоритета, с которым был принят пакет (для пакетов внутренней сигнализации может отсутствовать);

– значение max_ppdu_size: значение, получаемое от модуля Burst_packing через встроенный интерфейс. Отображает максимальное количество байт данных, которое сможет поместиться в будущий кадр.

Задачи модуля:

– определить, требуется ли фрагментация входному ALPDU пакету;

– в случае, если входному ALPDU пакету требуется фрагментация, то разделить его на части и задействовать один из алгоритмов контроля фрагментов;

– сформировать PPDU заголовок для входного пакета;

– инкапсулировать входной пакет в PPDU пакет;

– передать сформированный PPDU пакет в одну из очередей передатчика согласно его приоритету.

Формируемый формат данных: Payload-adapted PDU.

Его всевозможные структуры представлены на рисунке 11.

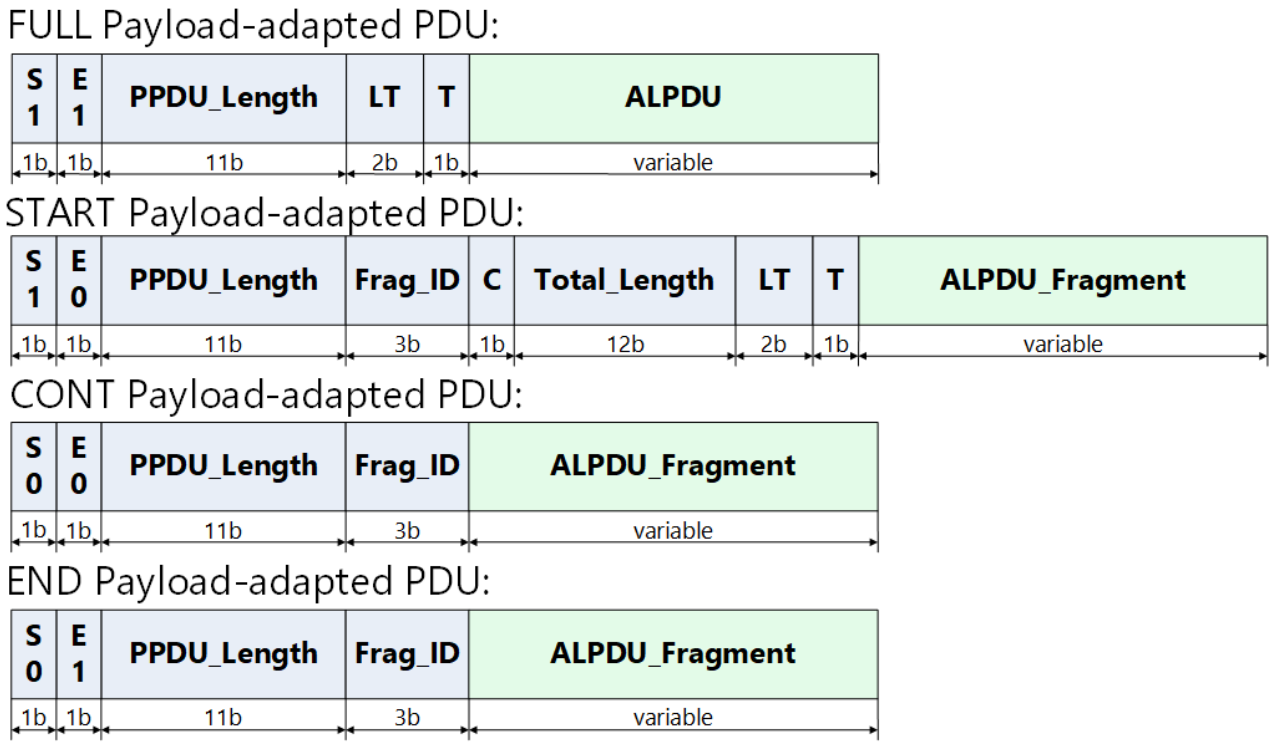


Рисунок 11 — Структуры всевозможных форматов PPDU

Данный формат предполагает использование 4 типов пакетов (сверху вниз):

• FULL PPDU: пакет PPDU, полностью вмещающий в себя пакет ALPDU. Используется в случае, если входной ALPDU пакет не фрагментируется;

• START PPDU: первый пакет PPDU в последовательности, содержит в себе первый фрагмент ALPDU пакета;

• CONTINUATION PPDU: PPDU пакеты следующие за предшествующим START PPDU пакетом, содержат в себе остальные фрагменты того же ALPDU пакета (кроме последнего);

• END PPDU: последний PPDU пакет в последовательности, содержит последний фрагмент того же ALPDU пакета.

Описание полей данного формата:

– поля S и E используются для определения типа пакета (одного из ранее описанных). Длина 2 бита;

– поле PPDU_Length содержит в себе длину PPDU пакета, без учёта первых двух байтов PPDU заголовка. В подсчёте длины не учитываются поля S, E, PPDU_Length и Frag_ID. Длина 11 бит;

– поле Frag_ID хранит в себе идентификатор PPDU пакета, он одинаков для всех PPDU, хранящих в себе фрагменты одного и того же ALPDU пакета. Не используется для типа пакета FULL PDU. Длина 3 бита;

– поле C указывает какой алгоритм защиты данных используется для пакета ALPDU, хранящемся в данном PPDU. Присутствует только в типе пакета START PPDU и зависит от значения поля use_alpdu_crc общей структуры параметров, описанной ранее. Длина 1 бит;

– поле Total_Length указывает размер всего ALPDU пакета, разделяющегося на фрагменты. Присутствует только в типе пакета START PPDU. Длина 12 бит;

– поле LT содержит в себе входное значение ALPDU_Label_size. Присутствует только в START PPDU и FULL PPDU. Длина 2 бита;

– поле T хранит в себе входное значение protocol_type_suppressed. Присутствует только в START PPDU и FULL PPDU. Длина 1 бит;

– поле ALPDU хранит в себе весь ALPDU пакет. Присутствует только в FULL PPDU. Длина варьируется;

– поле ALPDU_Fragment хранит в себе фрагмент ALPDU пакета. Присутствует во всех типах пакета PPDU, кроме FULL PPDU. Длина варьируется;

Данный модуль на основе входного значения max_ppdu_size определяет, нужно делить входной ALPDU пакет на части. Размер сформированного PPDU пакета (в байтах) обязательно должен быть меньше или равен этому значению. По этой причине сначала модуль проверяет помещается ли входной ALPDU пакет в FULL PPDU пакет и если это не так, то начинается процесс фрагментации.

Для процесса фрагментации пакетов данный модуль хранит в себе переменную Frag_ID, инициализируемую нулём. После фрагментации очередного ALPDU пакета данная переменная увеличивает своё значение на 1, при этом увеличение происходит по модулю 8, т.е. каждые 8 фрагментируемых ALPDU пакетов имеют уникальный Frag_ID при инкапсуляции в один из PPDU пакетов, содержащих эту переменную.

При фрагментации входного ALPDU пакета данный модуль использует один их двух алгоритмов для его защиты, чтобы из фрагментов в будущем можно было целостно собрать вложенный ALPDU пакет. Используемый алгоритм выбирается исходя из поля use_alpdu_crc общей структуры параметров.

В случае, если используется алгоритм sequence_number:

Данный модуль хранит переменные sequence_number, инициализируемые нулём, для каждого значения Frag_ID (в сумме 8). Перед фрагментацией данная переменная вставляется в конец фрагментируемого ALPDU пакета (т.е. после фрагментации будет занимать последний байт END PPDU пакета). После фрагментации очередного ALPDU пакета данная переменная увеличивает своё значение на 1, при этом увеличение происходит по модулю 256.

В случае, если используется алгоритм crc32:

Перед фрагментацией вычисляется crc32 значение для всего ALPDU пакета, а после заносится в конец фрагментируемого ALPDU пакета (т.е. после фрагментации будет занимать последние 4 байта END PPDU пакета).

Алгоритм sequence_number быстрее чем crc32, при этом использует меньше места в ALPDU пакете, но гарантирует только то, что в собранном после фрагментации пакете будут присутствовать все фрагменты в правильной последовательности. В свою очередь, алгоритм crc32 однозначно гарантирует что собранный пакет будет соответствовать исходному.

Выходные данные:

– сформированный PPDU пакет или пакеты, помещённые в одну из очередей передатчика, согласно входному значению QoS-тэга.

Алгоритм работы данного модуля представлен на рисунке 12.

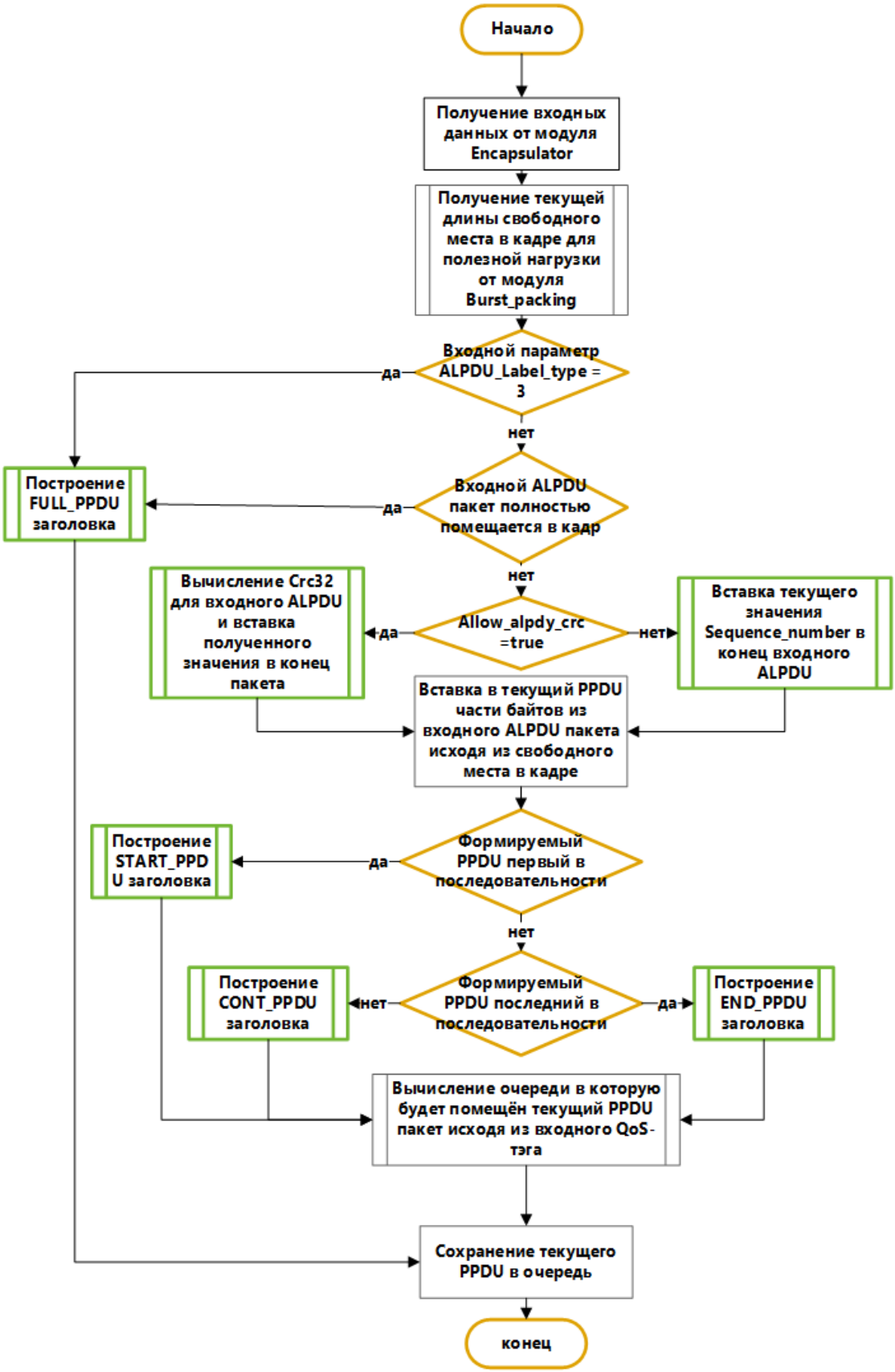


Рисунок 12 — Алгоритм работы модуля «Fragmenter»

2.2.3 Разработка модуля «Scheduler»

Алгоритм планирования: Строгий приоритет.

Задача модуля: передать ссылку на наиболее приоритетную очередь модулю «Burst packing», если она не пустая.

2.2.4 Разработка модуля «Burst packing»

Входные данные:

– пакет PPDU: пакет, полученный от модуля Scheduler;

– значение Frame_type: тип кадра, который будет сформирован. В совокупности с полем transmission_context_id общей структуры параметров влияет на формирование заголовка будущего Frame PDU кадра. Все принимаемые значения указаны в таблице 3;

– значение frame_size: значение, которое приходит с внешнего интерфейса и задаёт точную длину (в байтах) будущему кадру.

Таблица 3 – Возможные значения Frame_type

Frame_type	**Описание**
Logon	данный кадр используется для авторизации передатчика в спутниковой сети
Control	данный кадр используется для передачи различных сигнальных сообщений по спутниковому каналу связи
Traffic	данный кадр используется для передачи трафика по спутниковому каналу связи

Задачи модуля:

– сформировать заголовок кадра Frame PDU;

– инкапсулировать входные PPDU пакеты в кадр Frame PDU, если Frame_type::Trafifc;

– удалить ALPDU и PPDU заголовки, инкапсулировать PDU входного PPDU пакета в кадр Frame PDU, если Frame_type::Logon или Frame_type::Control;

– передать сформированный Frame PDU кадр интерфейсу физического уровеня, для дальнейшей его передачи по спутниковому каналу связи.

Формируемый формат: Frame PDU.

Его структуры представлены на рисунке 13.

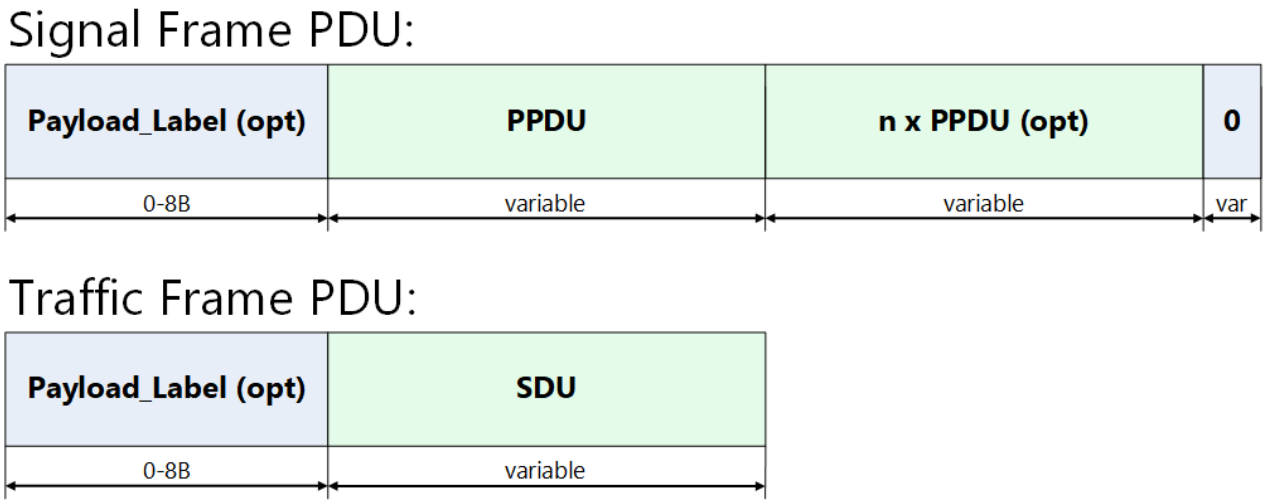


Рисунок 13 — Структуры форматов Frame PDU

Данный формат предполагает использование 2 типов кадров (сверху вниз):

• Signal Frame PDU: кадр Frame PDU, предназначенный для инкапсуляции одного пакета внутренней сигнализации спутниковой сети;

• Traffic Frame PDU: кадр Frame PDU, предназначенный для инкапсуляции одного или нескольких PPDU пакетов;

Описание полей структур данного формата:

– поле Payload_Label хранит в себе метку для адресации в сети. Это поле зависит от входного значения Frame_type и поля transmission_context_id общей

структуры параметров. Длина 0-8 байтов. Все возможные значения указаны в таблице 3;

– поле PPDU хранит в себе непосредственно один или несколько пакетов формата PPDU. Длина варьируется;

– поле SDU хранит в себе один пакет внутренней сигнализации спутниковой сети в его исходном виде. Длина варьируется;

– поле 0 хранит в себе байты заполнения, используется для дополнения полезной нагрузки (PPDU) до нужной длины кадра. Длина варьируется.

Таблица 3 — Возможные значения Payload_Label

Transmission Context	**Frame_type**		
	Logon	**Control**	**Traffic**
Transparent star, Dedicated access	Payload Label отсутствует	Payload Label отсутствует	Payload Label отсутствует
Transparent star, Slotted Aloha access	6 байтов; содержит RCST HID передатчика	3 байта; содержит значения Group ID, Logon ID	3 байта; содержит значения Group ID, Logon ID
Transparent star, CRDSA access	8 байтов; содержит RCST HID и CRDSA tag передатчика	5 байтов; содержит значения Group ID, Logon ID и CRDSA tag передатчика	5 байтов; содержит значения Group ID, Logon ID и CRDSA tag передатчика

Все необходимые значения для поля Payload_Label содержатся в структуре параметров передатчика, описанной ранее.

Выходные данные для передачи модулю «Fragmenter»:

– значение max_ppdu_size, содержащее доступный объём кадра для полезной нагрузки.

Выходные данные для передачи интерфейсу физического уровня:

– кадр Frame PDU, в последствии передаваемый интерфейсу физического уровень.

Алгоритм работы данного модуля представлен на рисунке 14.

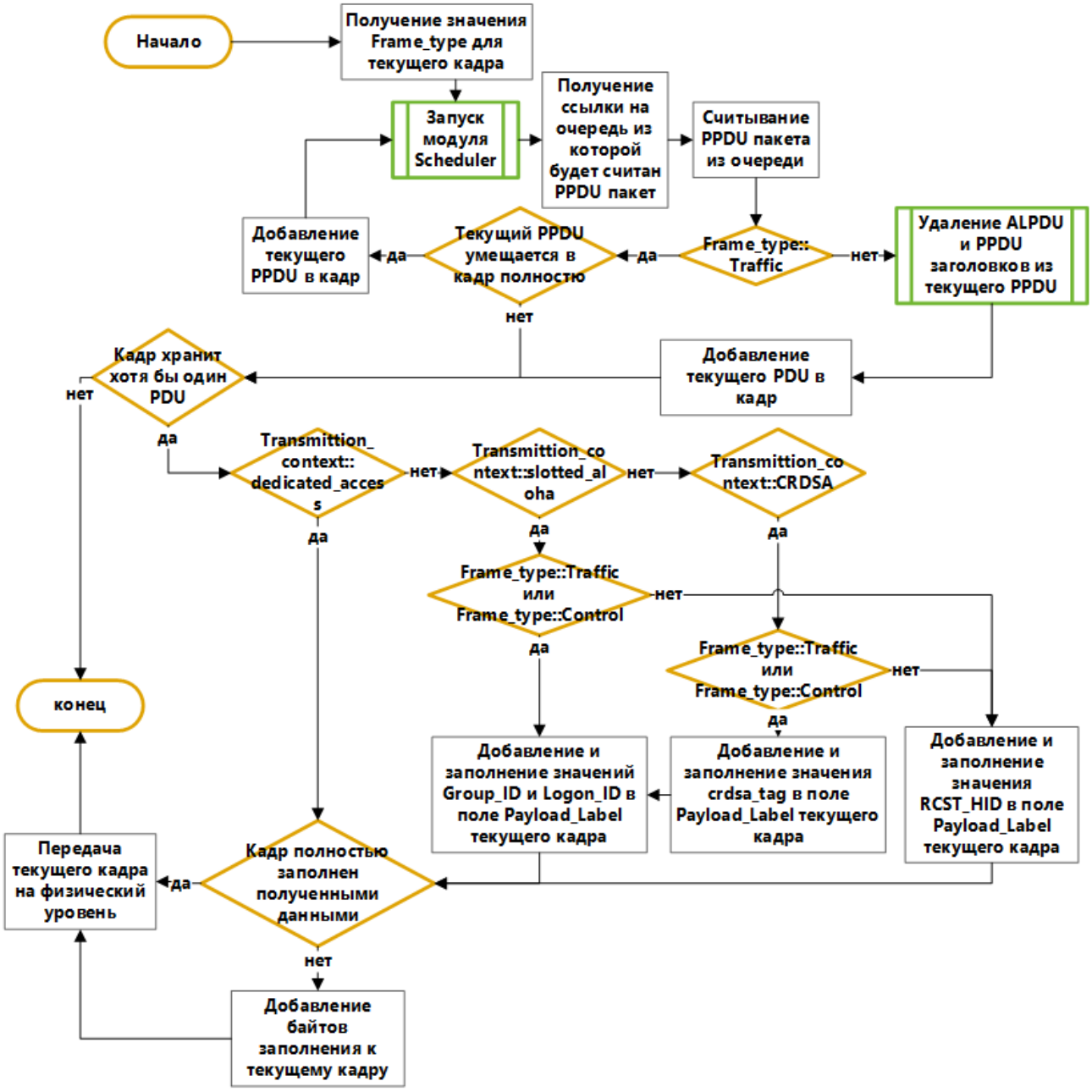


Рисунок 14 — Алгоритм работы модуля «Burst packing»

2.2.5 Диаграмма классов передатчика

Диаграмма классов передатчика представлена на рисунке 15.

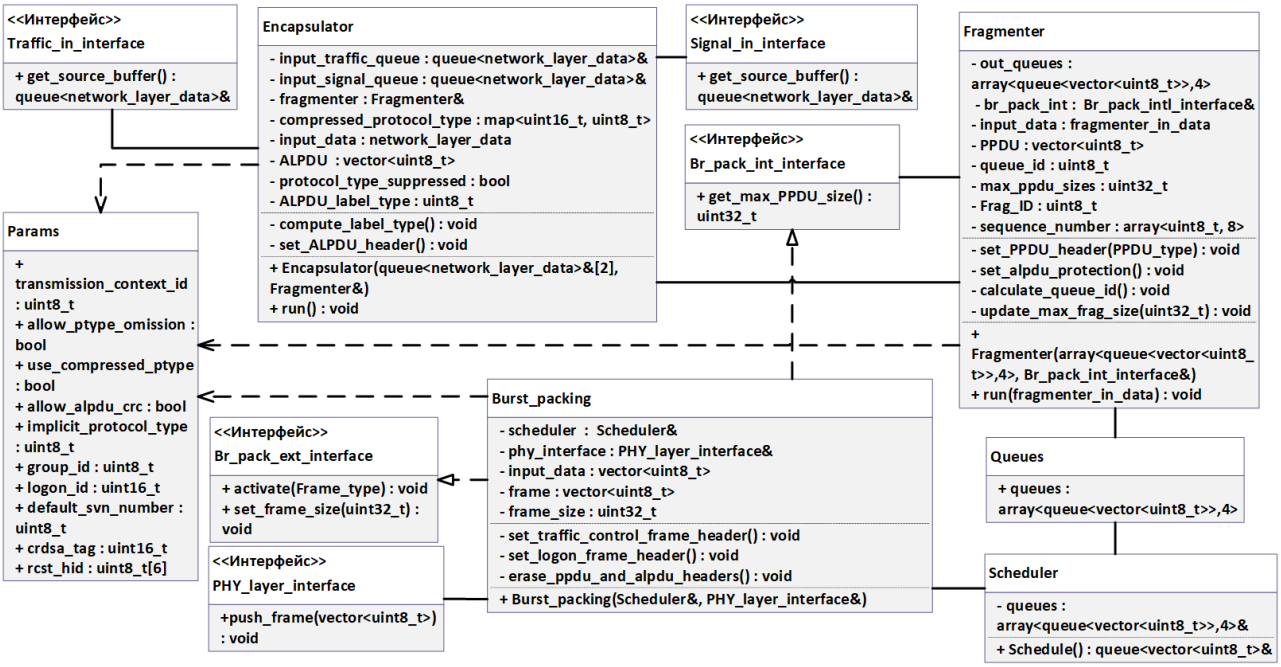


Рисунок 15 — Диаграмма классов передатчика

2.2.6 Диаграмма работы передатчика

Диаграмма работы передатчика представлена на рисунке 16.

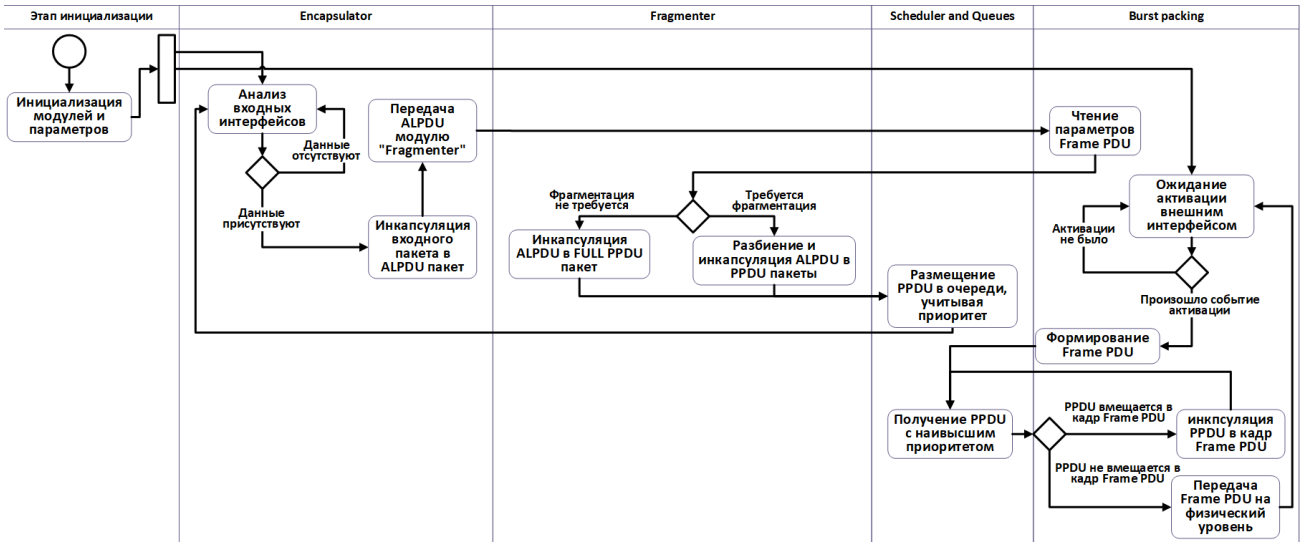


Рисунок 16 — Диаграмма работы передатчика

2.3 Разработка модулей приёмника

2.3.1 Разработка модуля «Burst unpacking»

Входные данные:

– кадр Frame PDU, полученный с интерфейса физического уровня;

– значение Frame_type: содержит тип входного кадра. Приёмник знает это значение исходя из контекста;

– значение Frame_size: содержит размер входного кадра. Приёмник знает это значение исходя из контекста.

Задачи модуля:

– получить кадр Frame PDU с физического уровня;

– исходя из входного значения Frame_type и значения поля transmission_context_id общей структуры параметров считать значения из поля Payload_Label Frame PDU заголовка;

– считать все PPDU пакеты из кадра Frame PDU если Frame_type::Traffic;

– считать PDU пакет из кадра Frame PDU если Frame_type::Logon или Frame_type::Control;

– передать модулю Defragmentation_engine все считанные PPDU пакеты, кроме пакетов с типом FULL PPDU. В случае, если Frame PDU заголовок включает в себя значения Group_ID и Logon_ID, передать модулю Defragmentation_engine вместе с PPDU пакетами;

– извлечь ALPDU пакеты из всех считанных PPDU пакетов, имеющих тип FULL PPDU, и передать их модулю «Decapsulator» вместе с полями LT и T из PPDU заголовка. В случае, если Frame PDU заголовок включает в себя значения Group_ID и Logon_ID, передать их модулю «Decapsulator» вместе с ALPDU пакетами;

– передать модулю «Decapsulator» считанный PDU пакет, если Frame_type::Logon или Frame_type::Control. В случае, если Frame PDU

заголовок включает в себя значения Group_ID и Logon_ID или RCST_HID, передать их модулю «Decapsulator» вместе с PDU пакетом.

Данный модуль принимает на входе кадр Frame PDU из физического канала связи и исследует его, считывая каждый бит. После извлечения Frame PDU заголовка, модуль исходя из входного значения Frame_type определяет какие данные инкапсулированы в текущий кадр Frame PDU. В последствии инкапсулированные данные извлекаются и для них определяется следующий модуль обработчик, «Defragmentation engine» или «Decapsulator».

Выходные данные для передачи модулю «Defragmentation engine»:

– пакет PPDU, содержащий фрагмент вложенного ALPDU;

– значения Group_ID и Logon_ID, если присутствуют в Frame PDU заголовке.

Выходные данные для передачи модулю «Decapsulator» при Frame_type::Traffic:

– пакет ALPDU, извлекаемый из PPDU пакета с типом FULL PPDU;

– значения LT и T, извлекаемые из заголовка PPDU пакета с типом FULL PPDU;

– значения Group_ID и Logon_ID, если присутствуют в Frame PDU заголовке.

Выходные данные для передачи модулю «Decapsulator» при Frame_type::Logon или Frame_type::Control:

– пакет PDU, извлекаемый из кадра Frame PDU;

– значения Group_ID и Logon_ID или RCST_HID, если присутствуют в Frame PDU заголовке.

Алгоритм работы модуля представлен на рисунке 17.

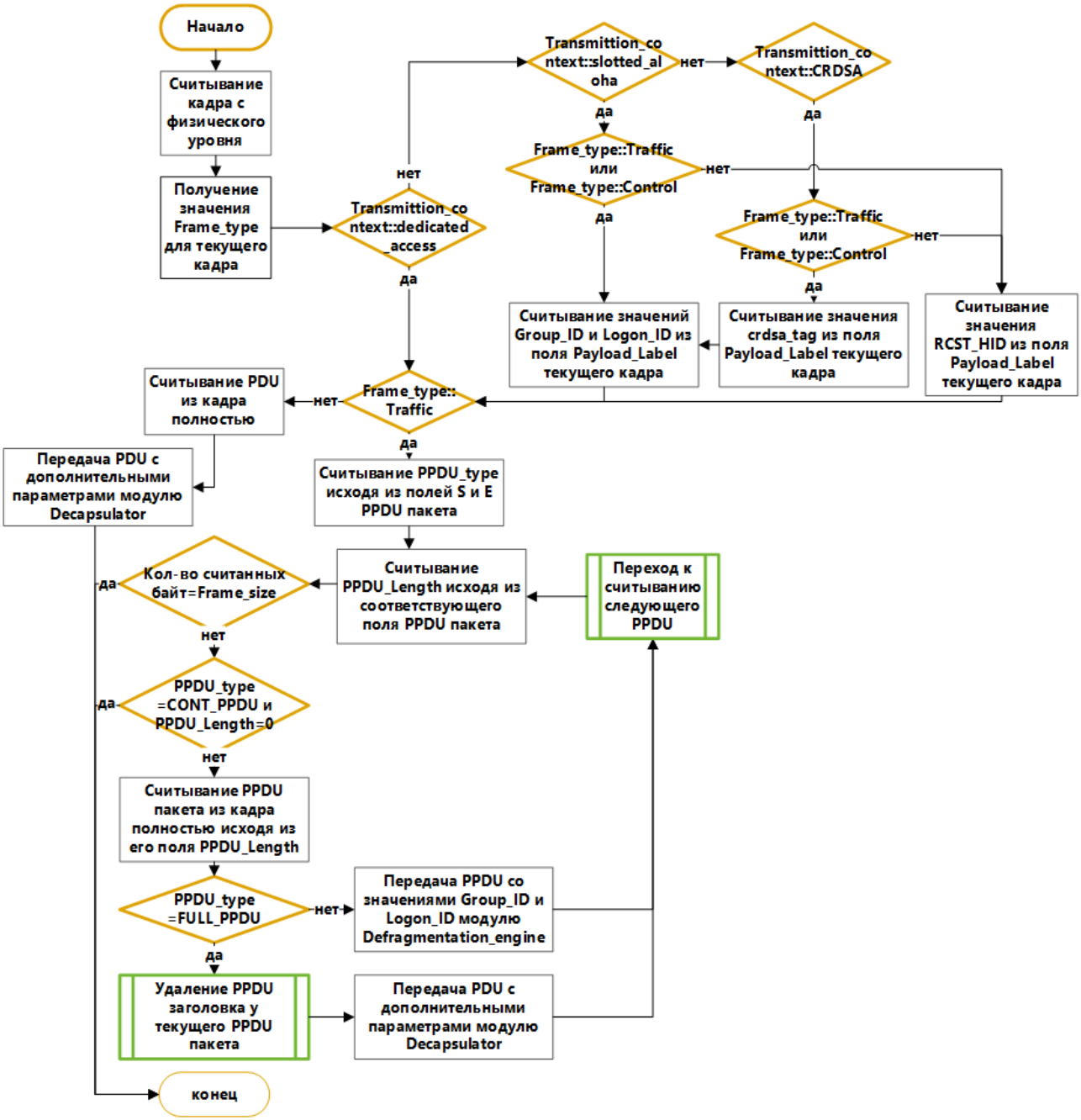


Рисунок 17 — Алгоритм работы модуля «Burst unpacking»

2.3.2 Разработка модуля «Defragmentation engine»

Входные данные:

– пакет PPDU, содержащий фрагмент вложенного ALPDU;

– значение Sender_ID: значение, содержащее объединённые воедино значения Group_ID и Logon_ID (если отсутствуют, значение инициализируется нулём).

Задачи модуля:

– получить PPDU пакет от модуля «Burst unpacking»;

– поместить входной PPDU пакет в буфер для текущих значений Frag_ID из PPDU заголовка и входного значения Sender_ID;

– извлечь из хранимых PPDU пакетов вложенный в них ALPDU пакет;

– проверить целостность извлечённого ALPDU пакета;

– отправить модулю «Decapsulator» извлечённый ALPDU пакет со значениями LT и T из PPDU заголовка, а также значение Sender_ID текущего буфера.

Данный модуль принимает на входе PPDU пакет и помещает его в один из своих буферов исходя из значений Sender_ID и Frag_ID текущего пакета. Для каждого пришедшего значения Sender_ID имеется свой буфер, включающий в себя ещё 8 буферов (Frag_ID_buffer) для каждого возможного значения Frag_ID. Каждый из 8 буферов хранит в себе PPDU пакеты, содержащие один и тот же ALPDU пакет, и значение sequence_number (инициализируется нулём), для проверки целостности ALPDU пакета после процесса сборки. Таким образом для отправителя (представленного значением Sender_ID) одновременно могут храниться фрагменты разных ALPDU пакетов (до 8).

Когда в один из таких буферов приходит PPDU пакет с типом END_PPDU начинается процесс сборки.

После сборки происходит процесс проверки целостности ALPDU пакета, за счёт одного из двух алгоритмов защиты: sequence_number и crc32.

Выходные данные:

– извлечённый ALPDU пакет;

– значения LT и T из PPDU заголовка первого PPDU в текущем буфере;

– значение Sender_ID текущего буфера;

Алгоритм работы модуля представлен на рисунке 18.

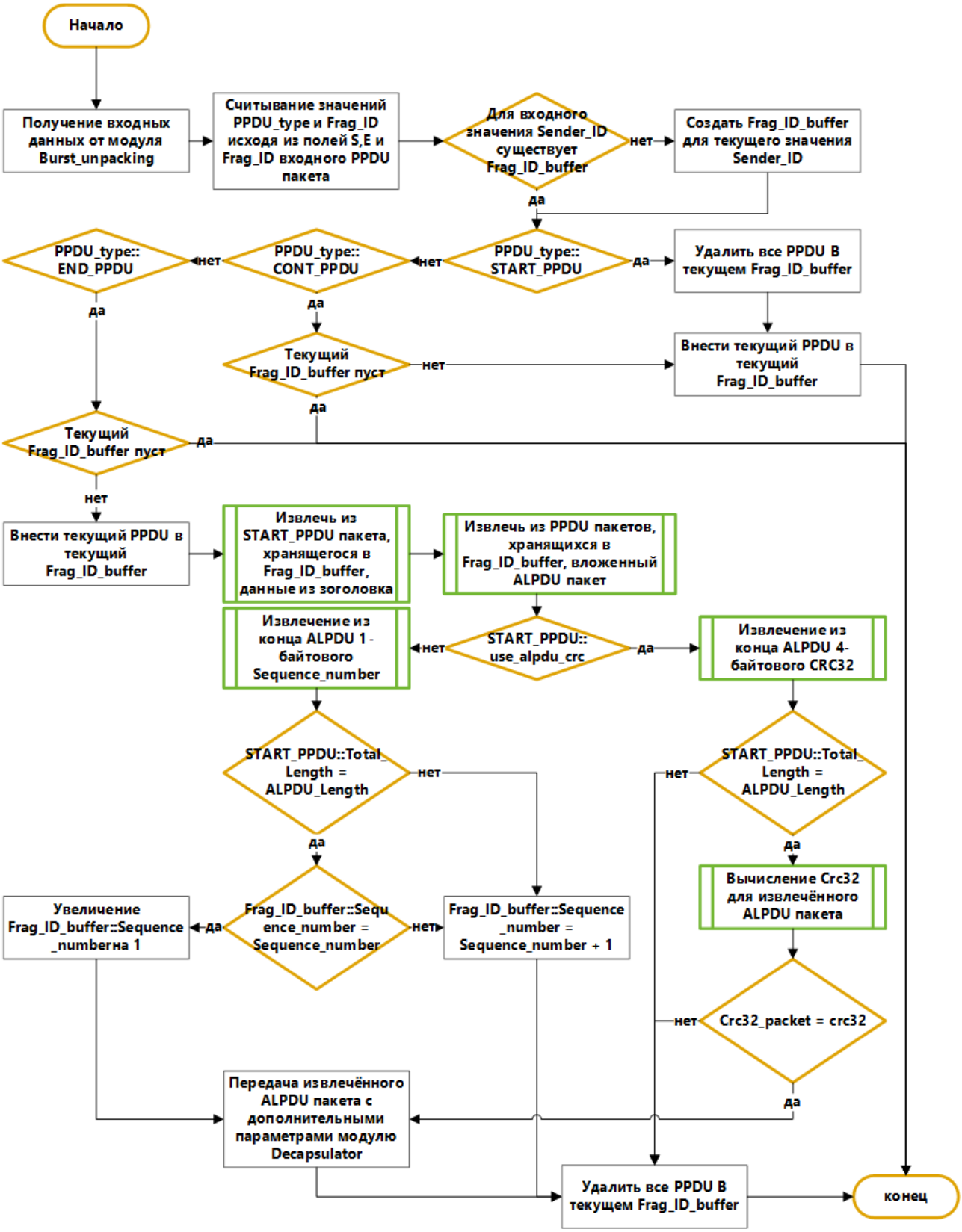


Рисунок 18 — Алгоритм работы модуля «Defragmentation engine»

2.3.3 Разработка модуля «Decapsulator»

Входные данные:

– пакет ALPDU или PDU, в зависимости от какого модуля он пришёл;

– значения ALPDU_Label_type и protocol_type_suppressed (если отсутствуют, принимают следующие значения: ALPDU_Label_type = 3, protocol_type_suppressed = true);

– значение Sender_ID (если отсутствует, принимает стандартное значение 0).

Задачи модуля:

– Получить входной ALPDU или PDU пакет от модулей «Defragmentation engine» и «Burst unpacking»;

– Извлечь из входного пакета исходный SDU пакет сетевого уровня или внутренней сигнализации;

– Передать извлечённый SDU пакет с дополнительными параметрами соответствующему интерфейсу (Signal_out_interface или Traffic_out_interface).

Выходные данные для интерфейса Traffic_out_interface:

– извлечённый SDU пакет сетевого уровня;

– извлечённый из ALPDU заголовка Protocol_type;

– входное значение Sender_ID;

Выходные данные для интерфейса Signal_out_interface:

– извлечённый SDU пакет внутренней сигнализации;

– входное значение Sender_ID;

Алгоритм работы модуля представлен на рисунке 19.

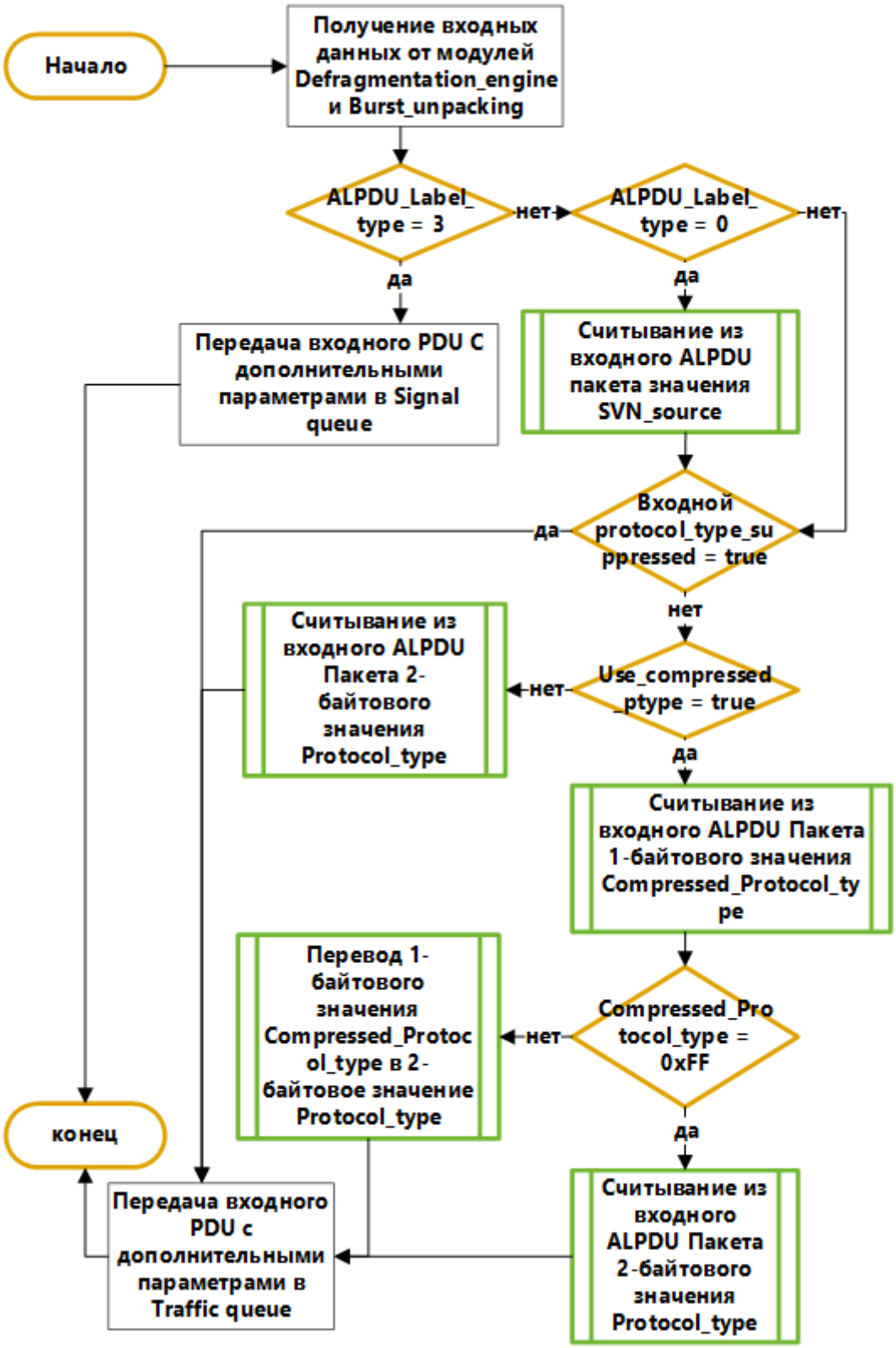


Рисунок 19 — Алгоритм работы модуля «Decapsulator»

2.3.4 Диаграмма классов приёмника

Диаграмма классов приёмника представлена на рисунке 20.

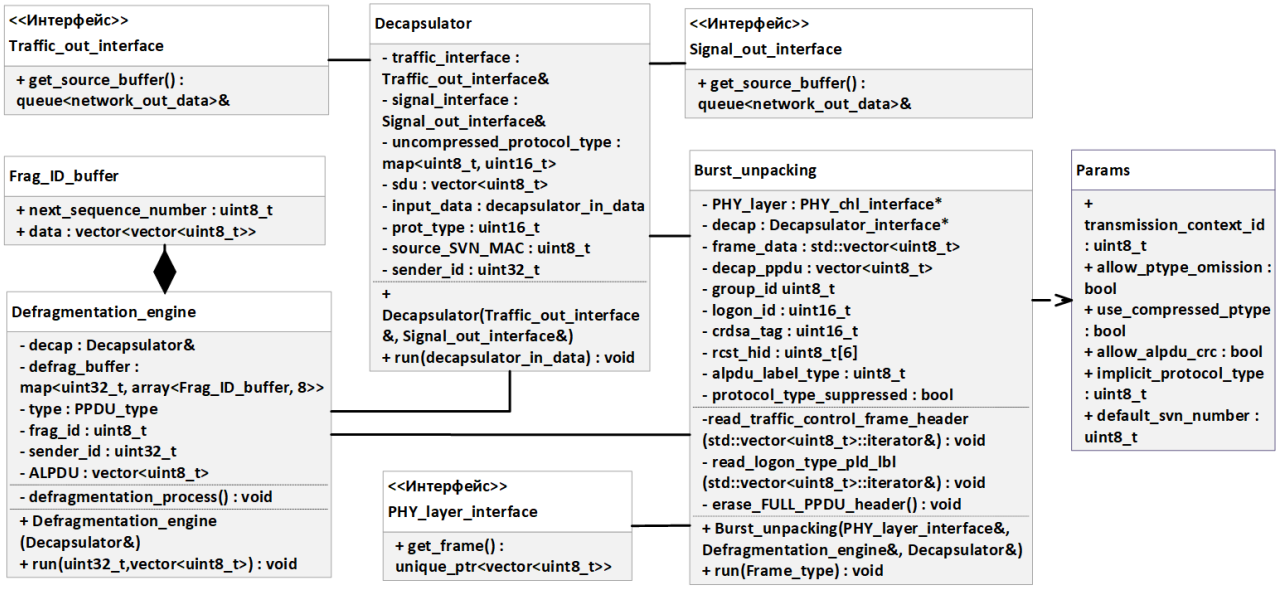


Рисунок 20 — Диаграмма классов приёмника

2.3.5 Диаграмма работы приёмника

Диаграмма работы приёмника представлена на рисунке 21.

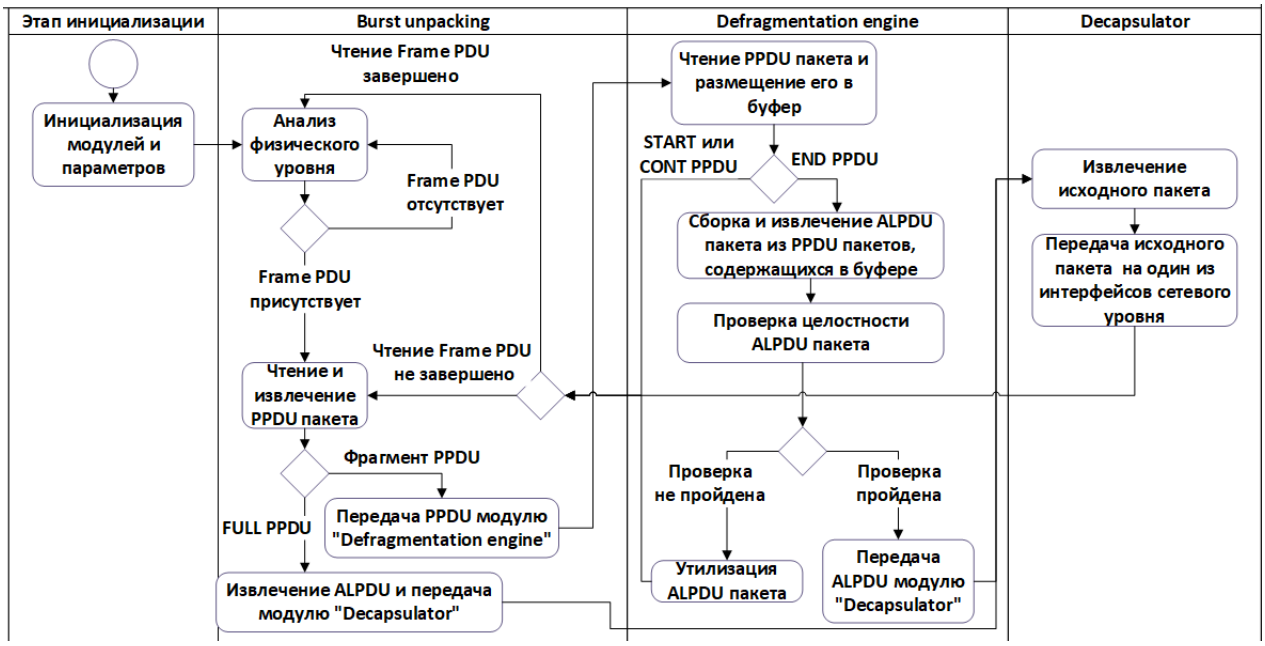


Рисунок 21 — Диаграмма работы приёмника

2.4 Выводы по второй главе

В ходе разработки модели были выявлены и описаны все составляющие модули, их задачи, а также входные и выходные данные. Для каждого из них был представлен алгоритм работы в виде блок-схемы. Были сформированы структурные схемы, диаграммы классов и работы передатчика и приёмника.

3 Реализация модели

Реализация включает в себя исходный код программы на языке C++ для каждого из ранее описанных модулей. Все модули представляют из себя классы.

3.1 Реализация модулей передатчика

3.1.1 Реализация модуля «Encapsulator»

Структура класса «Encapsulator» представлена на рисунке 22.

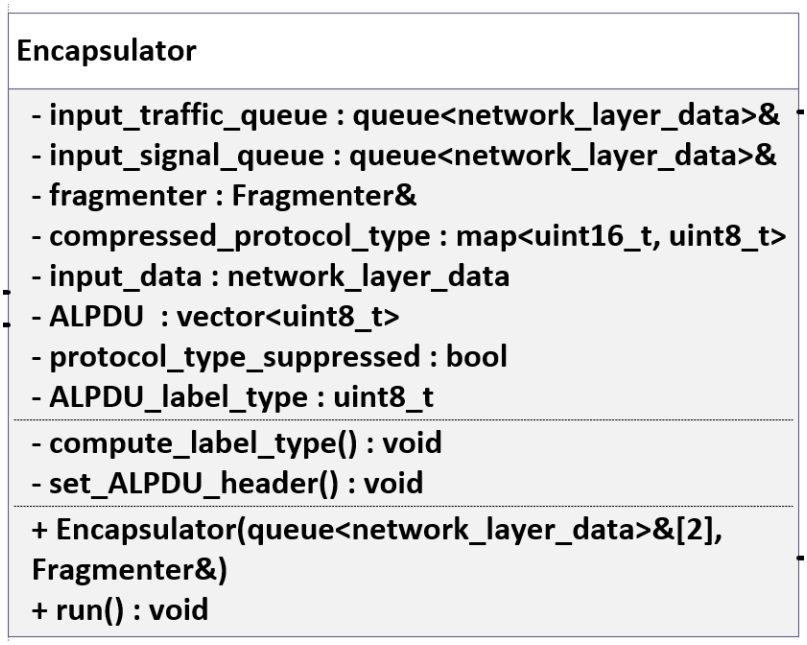


Рисунок 22 — Структура класса «Encapsulator»

Исходный код класса представлен в приложении А.

3.1.2 Реализация модуля «Fragmenter»

Структура класса «Fragmenter» представлена на рисунке 23.

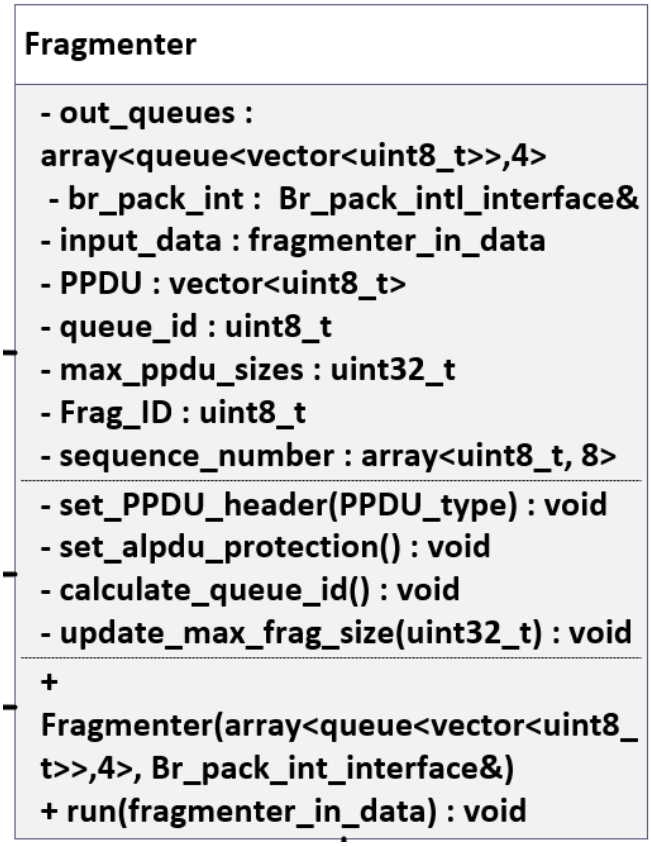


Рисунок 23 — Структура класса «Fragmenter»

Исходный код класса представлен в приложении Б.

3.1.3 Реализация модуля «Scheduler»

Структура класса «Scheduler» представлена на рисунке 24.

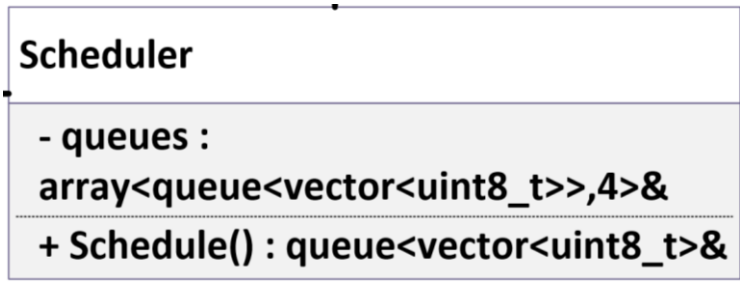


Рисунок 24 — Структура класса «Scheduler»

Исходный код класса представлен в приложении В.

3.1.4 Реализация модуля «Burst packing»

Структура класса «Burst_packing» представлена на рисунке 25.

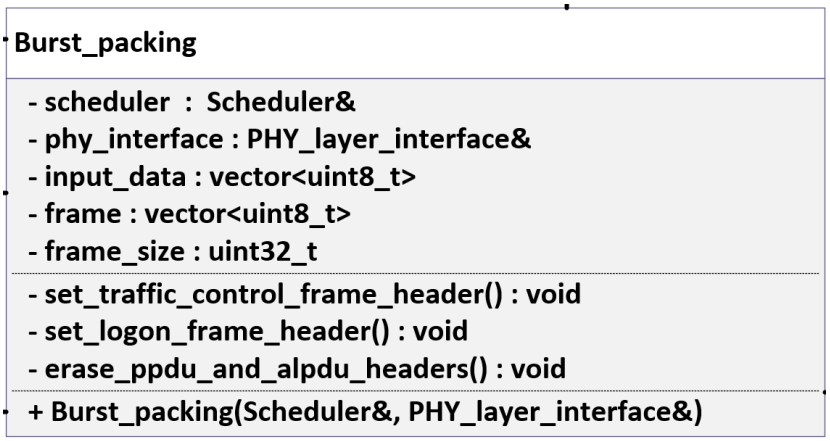


Рисунок 25 — Структура класса «Burst_packing»

Исходный код класса представлен в приложении Г.

3.2 Реализация модулей приёмника

3.2.1 Реализация модуля «Burst unpacking»

Структура класса «Burst_unpacking» представлена на рисунке 26.

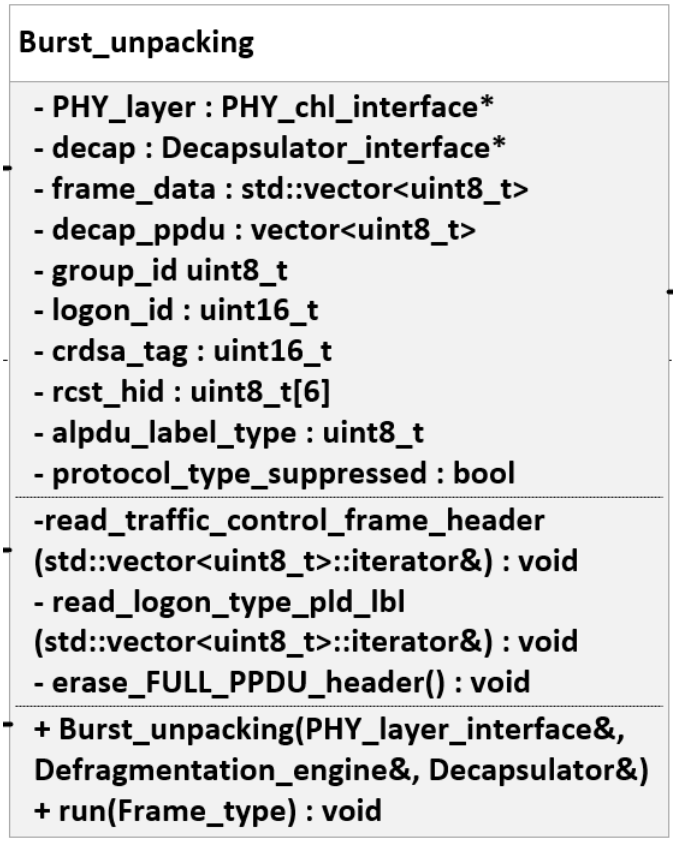


Рисунок 26 — Структура класса «Burst unpacking»

Исходный код класса представлен в приложении Д.

3.2.2 Реализация модуля «Fragmentation engine»

Структура класса «Fragmentation_engine» представлена на рисунке 27.

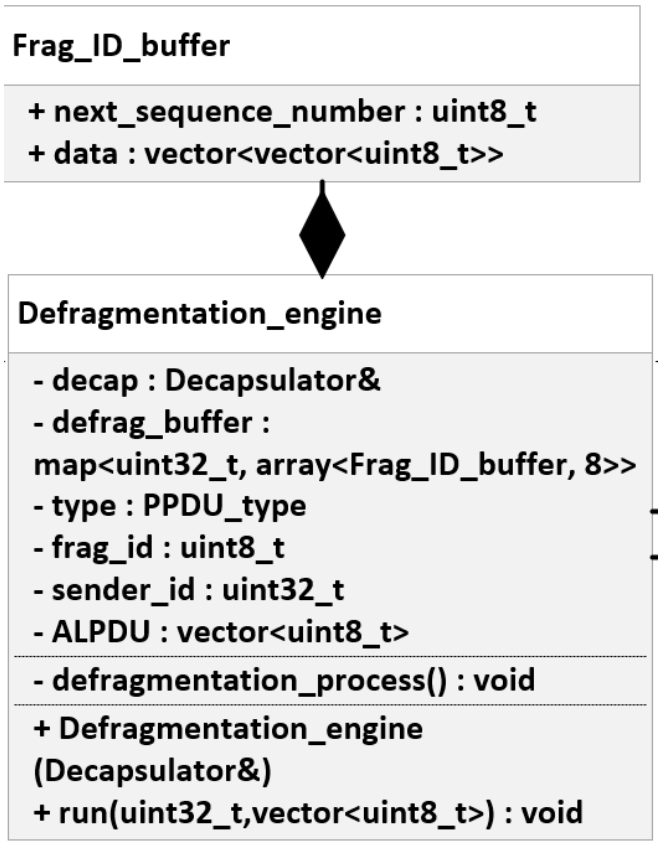


Рисунок 27 — Структура класса «Fragmentation engine»

Исходный код класса представлен в приложении Е.

3.2.3 Реализация модуля «Decapsulator»

Структура класса «Decapsulator» представлена на рисунке 28.

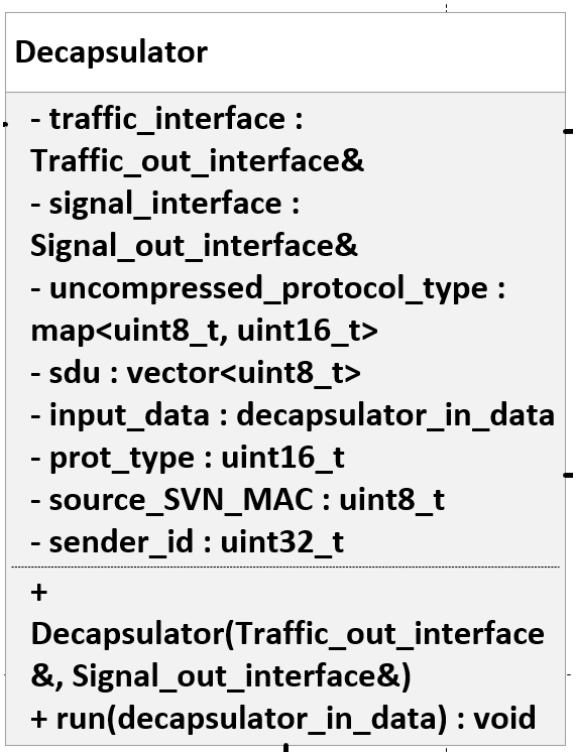


Рисунок 28 — Структура класса «Decapsulator»

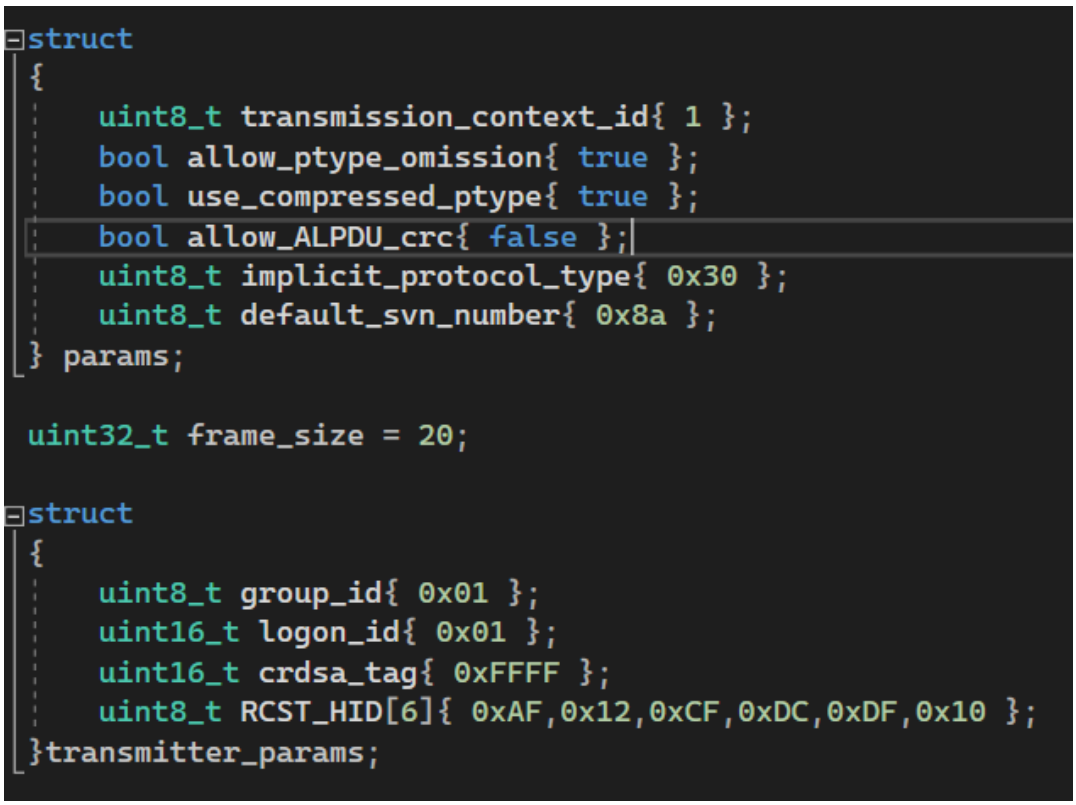
Исходный код класса представлен в приложении Ж.

3.3 Тестирование

Тестирование включает в себя в первую очередь сравнение входных пакетов, поступающих в передатчик, и выходных пакетов, выдающихся приёмником. Идентичность входных и выходных пакетов будет главным критерием оценки. Тем не менее, вывод программы отображает все пакеты, формируемые всеми модулями, поэтому можно увидеть, как они формируются.

Также в программе присутствует отдельный модуль, который выводит все входные и выходные пакеты, а также сравнивает их на идентичность.

Значения таблиц параметров для тестирования изображены на рисунке 29.



```
struct
{
    uint8_t transmission_context_id{ 1 };
    bool allow_ptype_omission{ true };
    bool use_compressed_ptype{ true };
    bool allow_ALPDU_crc{ false };
    uint8_t implicit_protocol_type{ 0x30 };
    uint8_t default_svn_number{ 0x8a };
} params;

uint32_t frame_size = 20;

struct
{
    uint8_t group_id{ 0x01 };
    uint16_t logon_id{ 0x01 };
    uint16_t crdsa_tag{ 0xFFFF };
    uint8_t RCST_HID[6]{ 0xAF,0x12,0xCF,0xDC,0xDF,0x10 };
}transmitter_params;
```

Рисунок 29 — Значения таблиц параметров

Значение Frame_size фиксировано для всех формируемых кадров Frame PDU.

3.3.1 Тестирование алгоритма фрагментации пакетов

На вход подаётся один IPv4 пакет размером в 50 байт. При текущих параметрах он должен быть разделён на 4 части и помещён в 4 кадра Frame PDU.

На рисунках 30 и 31 показан вывод программы.

Рисунок 30 — Вывод программы

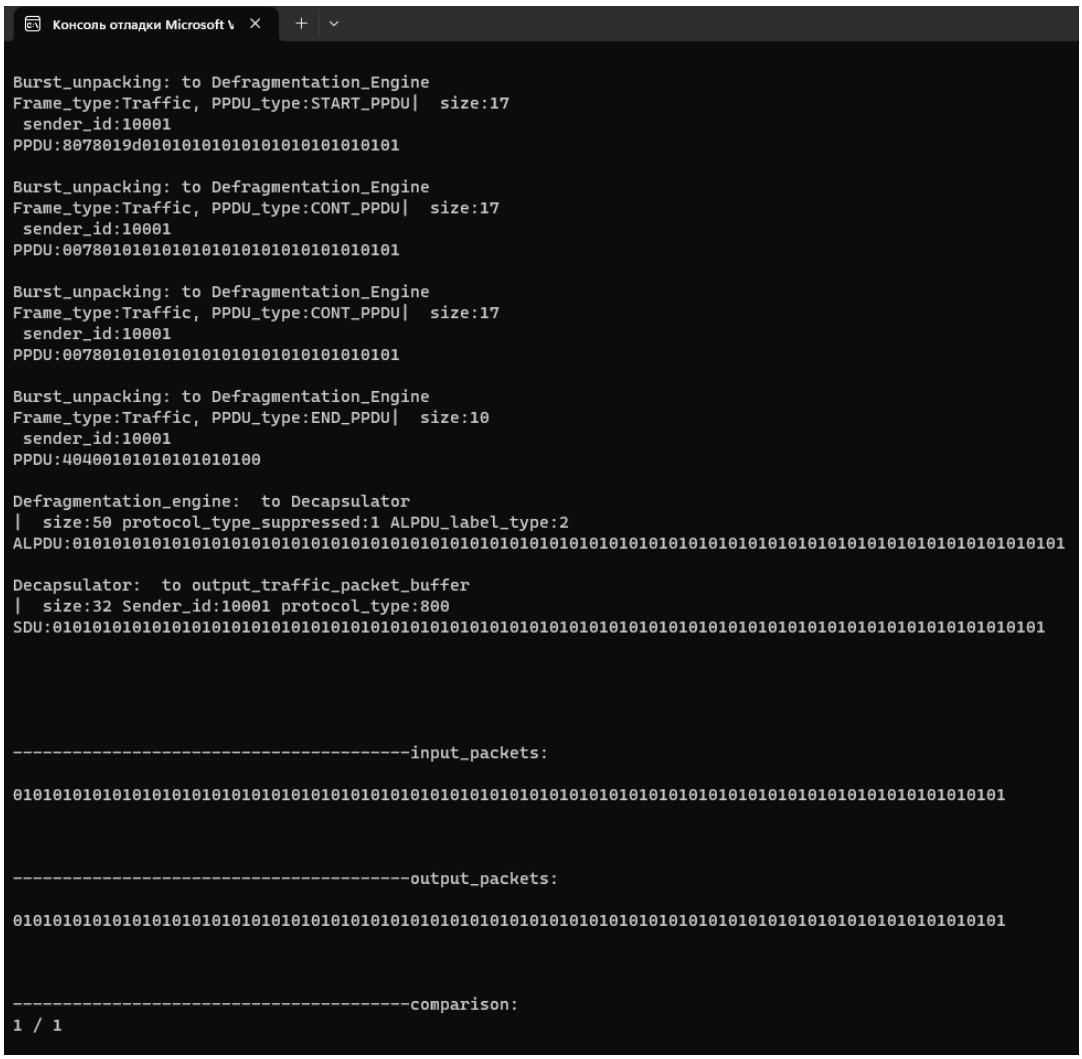


Рисунок 31 — Вывод программы

Как видно из выводов, входной пакет действительно был разделён на 4 части и уместился в 4 кадра Frame PDU, о чём говорит вывод модуля тестирования об идентичности пакетов. Тест пройден.

3.3.2 Тестирование алгоритма фрагментации пакетов при потере фрагментов

Параметры остаются теми же.

На рисунках 32 и 33 показан вывод программы.

Рисунок 32 — Вывод программы

Рисунок 33 — Вывод программы

Как видно из выводов, входной пакет действительно был разделён на 4 части и уместился в 4 кадра Frame PDU, но до модуля «Defragmentation Engine» не дошли два PPDU пакета с типом CONT_PPDU и по этой причине на выходе пакеты отсутствуют. Тест пройден.

3.3.3 Тестирование общего функционирования программы

Параметры остаются теми же.

На вход подаётся три IPv4 пакета и 2 пакета внутренней сигнализации. Все имеют собственный размер и приоритет.

На рисунках 34 и 35 показан вывод программы.

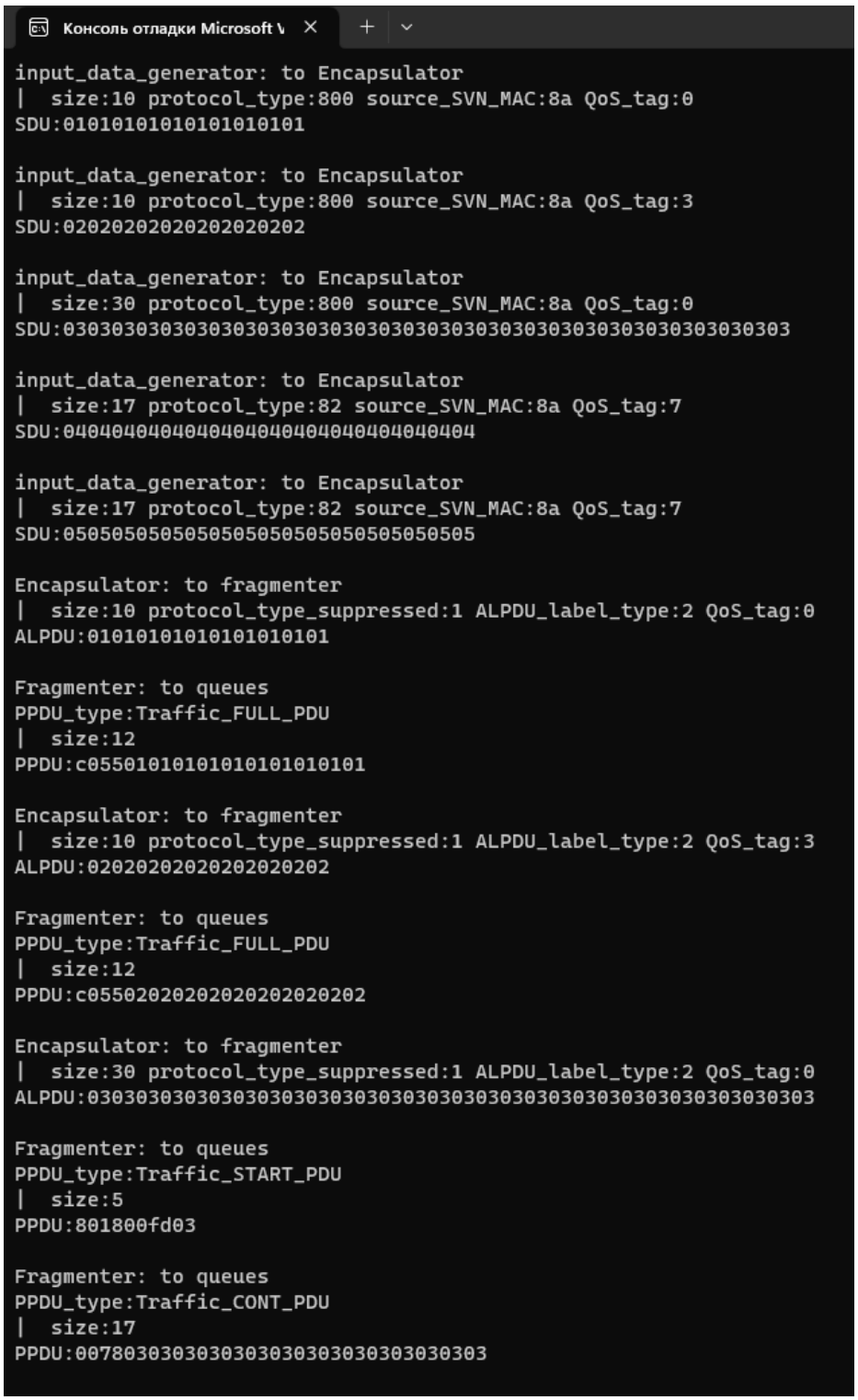


Рисунок 34 — Вывод программы

Рисунок 35 — Вывод программы

Как видно из выводов, все выходные пакеты соответствуют входным. Тест пройден.

3.3.4 Тестирование общего функционирования программы с новыми значениями параметров

Новые значения таблиц параметров для тестирования изображены на рисунке 36.

```
    struct
{
    uint8_t transmission_context_id{ 2 };
    bool allow_ptype_omission{ false };
    bool use_compressed_ptype{ false };
    bool allow_ALPDU_crc{ true };
    uint8_t implicit_protocol_type{ 0x30 };
    uint8_t default_svn_number{ 0x8a };
} params;

uint32_t frame_size = 25;

struct
{
    uint8_t group_id{ 0x01 };
    uint16_t logon_id{ 0x01 };
    uint16_t crdsa_tag{ 0xFFFF };
    uint8_t RCST_HID[6]{ 0xAF,0x12,0xCF,0xDC,0xDF,0x10 };
}transmitter_params;
```

Рисунок 36 — Значения таблиц параметров

На вход подаётся три IPv4 пакета и 2 пакета внутренней сигнализации. Все имеют собственный размер и приоритет.

На рисунках 37 и 38 показан вывод программы.

Рисунок 37 — Вывод программы

Рисунок 38 — Вывод программы

Как видно из выводов, с новыми параметрами все выходные пакеты всё ещё соответствуют входным. Тест пройден.

3.4 Выводы по третьей главе

Были реализованы все модули в виде программной модели, написанной на языке C++. Для каждого модуля был представлен исходный код и структуры классов. Было успешно проведено тестирование всего функционала программной модели.

ЗАКЛЮЧЕНИЕ

Разработана и реализована программная модель протокола канального уровня спутниковой системы. Разработанная модель протестирована на разных вариантах пакетов и параметрах протокола.

В будущем, эта модель будет использоваться как главный ориентир при реализации данного протокола под конкретное спутниковое оборудование.

СПИСОК СОКРАЩЕНИЙ

Protocol Data Unit (PDU) единица данных протокола,

Service Data Unit (SDU) единица служебных данных,

Addressed Link PDU (ALPDU) адресная единица данных протокола,

Payload-adapted PDU (PPDU) адаптированная к полезной нагрузке единица данных протокола,

Frame PDU единица данных протокола, представляющая кадр.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 DVB-RCS2 - Lower Layer Guidelines : сайт. – URL: https://dvb.org/?standard=guidelines-for-the-implementation-and-use-of-en-301-545-2-dvb-rcs2-lower-layer (дата обращения: 24.12.2022).

2 DVB-RCS2 Lower Layer Satellite Specification : сайт. – URL: https://dvb.org/?standard=dvb-rcs2-lower-layer-satellite-specification (дата обращения: 24.12.2022).

3 Satellite Earth Stations and Systems (SES); Return Link Encapsulation (RLE) protocol : сайт. – URL: https://www.etsi.org/deliver/etsi_ts/103100_103199/103179/01.01.01_60/ts_103179v010101p.pdf (дата обращения: 23.04.2023).

4 DVB-RCS2/S2 Testbed: A Distributed Testbed for Next-Generation Satellite System Design and Validation: сайт. — URL: https://elib.dlr.de/90560/1/06934571.pdf (дата обращения: 23.04.2023).

5 СТУ 7.5–07–2021 «Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности» – Текст: электронный.

ПРИЛОЖЕНИЕ А

Исходный код модуля «Encapsulator»

```
class Encapsulator
{
   std::queue<network_layer_data>& input_queue;
   Fragmenter& fragmenter;

   std::map<uint16_t, uint8_t> compressed_protocol_type{ {0x0082,0x42},{0x0800,0x30},{0x86dd,0x30} };

   network_layer_data input_data;

   std::vector<uint8_t> ALPDU;
   bool protocol_type_suppressed;
   uint8_t ALPDU_label_type;

   void compute_label_type();
   void set_ALPDU_header();

public:

   Encapsulator(std::queue<network_layer_data>& _input_queue, Fragmenter& _fragmenter);
   ~Encapsulator() = default;

   void run();
};
```

ПРИЛОЖЕНИЕ Б

Исходный код модуля «Fragmenter»

```
class Fragmenter
{
    std::array<std::queue<std::vector<uint8_t>>, 4>& out_queues;
    Fr_pack_internal_interface& fr_pack_func;

    std::array<uint32_t, 256> crc32_table;

    fragmenter_in_data input_data;
    std::vector<uint8_t> PPDU;

    uint32_t max_ppdu_size;
    std::array<uint32_t, 3> frame_free_space;

    uint8_t queue_id;

    uint8_t Frag_ID;
    std::array<uint8_t, 8> sequence_number;
    uint32_t crc32;

    void set_FULL_PPDU_header();
    void set_START_PPDU_header();
    void set_CONT_PPDU_header();
    void set_END_PPDU_header();
    void init_crc32_table();
    void set_alpdu_protection();
    void calculate_queue_id();
    void update_max_frag_size(uint32_t _max_ppdu_sizes);

public:
    Fragmenter(std::array<std::queue<std::vector<uint8_t>>, 4>& _out_queues, Fr_pack_internal_interface&
_fr_pack_func);
    ~Fragmenter() = default;

    void run(fragmenter_in_data);
};
```

ПРИЛОЖЕНИЕ В

Исходный код модуля «Scheduler»

```
class Scheduler
{
    std::array<std::queue<std::vector<uint8_t>>, 4>& scheduler_queues;

public:
    Scheduler(std::array<std::queue<std::vector<uint8_t>>, 4>& queues);
    std::queue<std::vector<uint8_t>>* schedule();
};

std::queue<std::vector<uint8_t>>* Scheduler::schedule()
{
    for (int i = 3; i >= 0; i--)
    {
        if (!scheduler_queues[i].empty())
        {
            std::queue<std::vector<uint8_t>>* queue_ptr = &scheduler_queues[i];
            return queue_ptr;
        }
    }
    return nullptr;
}
```

ПРИЛОЖЕНИЕ Г

Исходный код модуля «Burst packing»

```
class Burst_packing : public Fr_pack_internal_interface, public Fr_pack_external_interface
{
   Scheduler& scheduler;
   PHY_layer_interface& phy_interface;

   std::vector<uint8_t> input_data;

   std::vector<uint8_t> frame;
   uint32_t frame_size;

   void set_traffic_control_frame_header();
   void set_logon_frame_header();

   void erase_ppdu_and_ALPDU_headers();
   void run(Frame_type frame_type);

public:
   Burst_packing(Scheduler& _scheduler, PHY_layer_interface& _phy_interface);
   ~Burst_packing() = default;

   void activate(Frame_type) override;
   void set_frame_size(uint32_t) override;
   uint32_t get_max_PPDU_size() override;
};
```

ПРИЛОЖЕНИЕ Д

Исходный код модуля «Burst unpacking»

```
class Burst_unpacking
{
   PHY_layer_interface& phy_layer;
   Defragmentation_engine& def_engine;
   Decapsulator& decap;

   std::vector<uint8_t>::iterator frame_data_ptr;
   std::vector<uint8_t> decap_pdu;

   uint8_t group_id;
   uint16_t logon_id;
   uint16_t crdsa_tag;
   uint8_t rcst_hid[6];

   uint8_t ALPDU_label_type;
   bool protocol_type_suppressed;

   void read_traffic_control_frame_header(std::vector<uint8_t>::iterator&);
   void read_logon_frame_header(std::vector<uint8_t>::iterator&);

   void erase_FULL_PPDU_header();

public:
   Burst_unpacking(PHY_layer_interface&, Defragmentation_engine&, Decapsulator&);
   ~Burst_unpacking();
   void run(Frame_type);
};
```

ПРИЛОЖЕНИЕ Е

Исходный код модуля «Defragmentation engine»

```
class Defragmentation_engine
{
   struct Frag_ID_buffer
   {
      uint16_t total_Length;
      uint8_t label_type;
      bool type_suppressed_flag;
      bool Use_Packet_CRC;
      uint8_t next_sequence_number{ 0 };
      std::vector<std::vector<uint8_t>> data;
   };

   Decapsulator& decap;

   std::array<uint32_t, 256> crc32_table;
   std::map<uint32_t, std::array<Frag_ID_buffer, 8>> defrag_buffer;

   PPDU_type type;
   uint8_t frag_id;
   uint32_t sender_id;
   uint8_t sequence_number;
   uint32_t crc32_packet;
   uint32_t crc32;

   std::vector<uint8_t> ALPDU;
   void defragmentation_process();
   void init_crc32_table();

public:
   Defragmentation_engine(Decapsulator&);
   ~Defragmentation_engine() = default;

   void run(uint32_t sender_id, std::vector<uint8_t> ppdu);

};
```

ПРИЛОЖЕНИЕ Ж

Исходный код модуля «Decapsulator»

```
class Decapsulator
{

    Traffic_out_interface& traffic_interface;
    Signal_out_interface& signal_interface;

    std::map<uint8_t, uint16_t> uncompressed_protocol_type{ {0x42,0x0082},{0x30,0x0800} };

    //decapsulator_in_data input_data;

    std::vector<uint8_t> sdu;

    uint16_t prot_type;
    uint8_t source_SVN_MAC;

public:
    Decapsulator(Traffic_out_interface&, Signal_out_interface&);
    ~Decapsulator() = default;
    void run(decapsulator_in_data input_data);
};
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«23» 06 2023 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Моделирование и прототипирование протокола канального уровня спутниковой системы
тема

Руководитель _____ 23.06.23 старший преподаватель И.Н.Рыженко
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 23.06.23 А.А.Коряпин
подпись, дата инициалы, фамилия

Нормоконтролер _____ 23.06.23 старший преподаватель И.Н.Рыженко
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2023