НАЦИОНАЛЬНЫЙ СТАНДАРТ РЕСПУБЛИКИ КАЗАХСТАН

Информационные технологии

ИНТЕРНЕТ ВЕЩЕЙ

Протокол беспроводной передачи данных на основе узкополосной модуляции радиосигнала (NB-Fi)

Information Technology. Internet of things. Wireless protocol based on narrow band RF modulation (NB-Fi)

Дата введения – – –

1 Область применения

В настоящем стандарте установлены требования к протоколу обмена для интернета вещей в узкополосном спектре (NB-Fi), включая требования к:

- физическому уровню (раздел 5);
- MAC-уровню (раздел 6);
- транспортному уровню (раздел 7).

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

СТ РК 3105-2017 Сети последующих поколений. Структура и функциональные модели архитектуры. Термины и определения для интернета вещей.

СТ РК ISO/TS 37151-2016 Интеллектуальные инфраструктуры коммунального хозяйства Принципы и требования к системе рабочих показателей

Примечание - При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов и классификаторов по ежегодно издаваемому информационному указателю «Нормативные документы по стандартизации» по состоянию на текущий год и соответствующим ежемесячно издаваемым информационным указателям, опубликованным в текущем году. Если ссылочный документ заменен (изменен), то при пользовании настоящим Стандартом следует руководствоваться замененным (измененным) документом. Если ссылочный документ отменен без замены, то положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **радиотрансивер**: Интегральная схема, предназначенная для приема/передачи данных с использованием радиосигналов (в том числе посредством протокола NB-Fi).

3.2 **устройство приема-передачи данных/модем**: Программно-аппаратный комплекс со встроенным радиотрансивером, являющийся либо самостоятельным оборудованием, либо встроенным компонентом в оконечное устройство, применяемый для приема или передачи данных с использованием радиотрансивера.

3.3 **оконечное устройство**: Оборудование, содержащее устройство приема/передачи данных (модем).

3.4 **базовая станция NB-Fi**: Программно-аппаратный комплекс, обеспечивающий прием и передачу данных посредством радиосигналов от оконечных устройств, работающих по протоколу NB-Fi с одной стороны и взаимодействие с сервером NB-Fi с использованием широкополосного канала связи с другой стороны.

3.5 **сервер NB-Fi**: Программно-аппаратный комплекс, являющийся центральным узлом, выполняющим информационное взаимодействие со множеством оконечных устройств по протоколу транспортного уровня NB-Fi, с использованием распределенной сети базовых станций NB-Fi.

3.6 **пакет восходящего направления (UPLINK-пакет)**: Пакет данных, передаваемый устройствами и принимаемый базовыми станциями.

3.7 **пакет нисходящего направления (DOWNLINK-пакет)**: Пакет данных, передаваемый передатчиком базовой станции и принимаемый устройствами.

4 Сокращения

В настоящем стандарте применены следующие сокращения:

ОФМн-2 – относительная двоичная фазовая манипуляция несущей;

ЧМн – частотная манипуляция несущей;

«Магма» – алгоритм симметричного блочного шифрования согласно ГОСТ Р 34.12-2015;

LBT – Listen Before Talk, режим прослушивания перед излучением;

CRC – Cyclic Redundancy Check, циклический избыточный код;

LPWAN – Low-power Wide-area Network, энергоэффективные сети связи дальнего радиуса действия.

5 Физический уровень (Physical layer)

5.1 Общие положения

Физический уровень обеспечивает механизм приема/передачи произвольной информации по среде распространения. В данном разделе установлены требования к техническим характеристикам физического уровня для двух типов пакетов данных:

- UPLINK-пакет (см. 0);
- DOWNLINK-пакет (см. 0).

5.2 UPLINK-пакет

UPLINK-пакет представляет собой модулированные последовательности двоичных данных, сгруппированных в байты.

Описание UPLINK-пакета приведено в таблице 1. Значения параметров, приведенных в таблице 1, определены для диапазона рабочих температур от минус 40 °С до плюс 70 °С.

Таблица 1 – Основные технические характеристики UPLINK-пакета

Наименование параметра	Значение (характеристика) параметра
Минимальная ширина полосы рабочих частот приема базовой станции	51,2 кГц
Скорость передачи данных	50, 400, 3200, 25600 бит/с
Длина пакета	288 бит
Модуляция	ОФМн-2
Предельная чувствительность приема для скорости передачи данных, бит/с: - 50 - 400 - 3200 - 25600	 Минус 150 дБм Минус 141 дБм Минус 132 дБм Минус 123 дБм
Метод разделения каналов	Частотный
Количество одновременно принимаемых каналов при скорости 50 бит/с и ширине полосы частот 51,2 кГц	1024
Количество одновременно принимаемых каналов при скорости 400 бит/с и ширине полосы частот 51,2 кГц	128
Количество одновременно принимаемых каналов при скорости 3200 бит/с и ширине полосы частот 51,2 кГц	16
Количество одновременно принимаемых каналов при скорости 25600 бит/с и ширине полосы частот 51,2 кГц	1

Наименование параметра	Значение (характеристика) параметра
Диапазон перестройки выходной мощности передатчика, дБ не менее	24
Шаг перестройки выходной мощности передатчика, дБ не более	3
Минимальная пропускная способность приема UPLINK-пакетов одной базовой станцией	20 МБт/сут

Примечания

1 Ввиду малых значений ширины полосы сигналов используется относительная фазовая манипуляция с целью минимизации влияния ухода частоты опорного генератора за время отправки пакета. Для самой низкой скорости передачи (50 бит/с) время отправки 1 бита данных будет составлять 20 мс. Необходимую стабильность частоты обеспечивают кварцевые осцилляторы с температурной нестабильностью не более 0,5 ppm.

2 ОФМн-2 с низкой скоростью передачи битов данных может быть сформирована на аппаратном уровне не всеми радиотрансиверами. Для формирования данного вида модуляции может быть использована ЧМн с более высокой скоростью передачи битов данных.

3 Мощность тепловых шумов в полосе 50 Гц при температуре 290 К составляет:

$$N = -174 + 10 \times \log_{10}50 = -157 \text{ дБм}$$

При входном коэффициенте шума базовой станции, равном 2 дБ, а также отношении сигнал/шум (Signal to Noise Ratio, SNR), равном 5 дБ, при котором достигается частота ошибок по битам (Bit Error Rate, BER) BER = 10^{-5} вычисляют предельную теоретическую чувствительность S, равную:

$$S = -157 + 2 + 5 = -150 \text{ дБм}.$$

4 Ширина рабочей полосы частот приема базовой станцией должна быть не менее значения, указанного в таблице. В данной полосе частот должен осуществляться прием одновременно всех скоростей передачи данных на всех каналах, количество которых для каждой скорости указано в таблице.

5 Частота передачи сигналов должна определяться псевдослучайным образом в пределах рабочей полосы частот в зависимости от ряда параметров. Алгоритм определения частоты передачи сигналов UPLINK-пакетов приведен в разделе А.1 приложения А.

5.3 DOWNLINK-пакет

DOWNLINK-пакет представляет собой модулированные последовательности двоичных данных, сгруппированных в байты.

Описание DOWNLINK-пакета приведено в таблице 2. Значения параметров, приведенных в таблице 2, определены для диапазона рабочих температур от минус 40 °С до плюс 70 °С.

Таблица 2 – Основные технические характеристики DOWNLINK-пакета

Наименование параметра	Значение (характеристика) параметра
Минимальная ширина полосы рабочих частот передачи базовой станции	102,4 кГц
Скорость передачи данных	50 (опционально), 400, 3200, 25600 бит/с
Модуляция	ОФМн-2 или ФМн-2
Предельная чувствительность приема для скорости передачи данных, бит/с: - 50 - 400 - 3200 - 25600	 Минус 148 дБм Минус 139 дБм Минус 130 дБм Минус 121 дБм
Метод разделения каналов	Частотный
Диапазон перестройки выходной мощности передатчика, дБ не менее	24
Шаг перестройки выходной мощности передатчика, дБ не более	3
Минимальная пропускная способность передачи DOWNLINK-пакетов одной базовой станцией	10 МБт/сут (при условии работы 100 % устройств на скорости DOWNLINK 25600 бит/с)

Примечания

1 При использовании скоростей 50 и 400 бит/с, неточность формирования частоты задающим генератором может приводить к возникновению проблем, связанных с несовпадением частот передатчика и приемника, что вызывает потери пакетов. Для решения этой проблемы необходимо применять алгоритм компенсации нестабильности частот задающих генераторов. Описание алгоритма компенсации нестабильности частот задающего генератора приведено приложении Б.

2 Ширина рабочей полосы частот передачи базовой станцией должна быть не менее значения, указанного в таблице. В данной полосе частот должна осуществляться передача по крайней мере одного канала с возможностью произвольного выбора скорости передачи и частоты передачи в пределах рабочей полосы частот.

3 Частота передачи сигналов определяется псевдослучайным образом в пределах рабочей полосы частот в зависимости от ряда параметров. Алгоритм определения частоты передачи сигналов DOWNLINK-пакетов приведен в разделе А.2 приложения А.

5.4 Режим работы LBT (Listen Before Talk)

При включенном режиме работы LBT (Listen Before Talk – режим прослушивания перед излучением) устройство, прежде чем выполнить переход в режим передачи для отправки пакетов, должно переключиться в режим оценки уровня сигнала в полосе частот, в которой предполагается отправка данных. Если уровень сигнала не превышает

установленного значения, что означает отсутствие передачи другим устройством, то переход в режим передачи и отправка данных могут быть выполнены. В противном случае устройство не должно выполнять передачу до тех пор, пока уровень сигнала в данной полосе частот не упадет ниже установленного значения. Для включения/выключения режима работы LBT необходимо использовать конфигурационный флаг транспортного уровня FLG_LBT [см. 0, перечисление p)].

6 MAC-уровень (Media Access Control layer, MAC)

6.1 Общие положения

MAC-уровень обеспечивает передачу датаграммы (информационного пакета) на выбранном физическом уровне и описывает следующие параметры:

- формат полей пакета;
- способы адресации;
- методы защиты данных;
- методы контроля целостности данных;
- методы восстановления ошибок.

Описание MAC-уровня приведено в таблице 3. Значения параметров, приведенных в таблице 3, определены для диапазона рабочих температур от минус 40 °С до плюс 70 °С.

Таблица 3 – Основные технические характеристики MAC-уровня

Наименование параметра	Значение (характеристика) параметра
Номерная емкость сети	4294967296 (2^{32}) устройств
Эффективные скорости передачи данных (UPLINK-пакет)	12,5, 100, 800, 6400 бит/с
Виды помехоустойчивого кодирования (UPLINK-пакет)	Несистематический сверточный код, несистематический полярный код
Скорость помехоустойчивого кодирования (UPLINK-пакет)	5/8
Длина полезных данных одного пакета (UPLINK-пакет)	9 байт
Эффективные скорости передачи данных (DOWNLINK-пакет)	12,5, 100, 800, 6400 бит/с
Вид помехоустойчивого кодирования (DOWNLINK-пакет)	ZIGZAG код
Скорость помехоустойчивого кодирования (DOWNLINK-пакет)	1/2
Длина полезных данных одного пакета (DOWNLINK-пакет)	9 байт

Алгоритм шифрования полезных данных	«Магма»
Используемый размер ключа	256т

Примечание – Вместо алгоритма шифрования «Магма» возможно использование другого алгоритма блочного симметричного шифрования со следующими параметрами: размер блока данных – 64 бит, размер ключа – 256 бит, при наличии достаточных оснований для этого. При этом использование других алгоритмов, не являющихся российским стандартом, может накладывать ограничения на сферу применения настоящего стандарта.

6.2 UPLINK-пакет

Общая длина UPLINK-пакета должна составлять 36 байт. Размер поля Payload (данные транспортного уровня) должен составлять 9 байт. Программный код реализации функции формирования UPLINK-пакета на языке Си приведен в разделе Е.1 приложения Е.

Формат UPLINK-пакетов одинаков для различных скоростей передачи данных. Порядок следования битов в байтах UPLINK-пакета – от старшего к младшему.

Структура формата UPLINK-пакета приведена в таблице 4.

Таблица 4 – Структура формата UPLINK-пакета

<table>
<tr><td colspan="4">Preamble (Преамбула) (4 байта)</td><td colspan="8">Error correction code (Помехоустойчивый код) (32 байта)</td></tr>
<tr><td colspan="4" rowspan="2">Preamble (Преамбула)</td><td colspan="8">Error correction code source (Входные данные для кодера помехоустойчивого кода) (20 байт)</td></tr>
<tr><td colspan="4">Modem_ID (Идентификатор, присвоенный устройству)</td><td>Crypto Iter (Криптоитератор)</td><td>Payload (Данные транспортного уровня)</td><td>MIC0_7 (Имитовставка)</td><td>Packet CRC (Контрольная сумма пакета данных)</td></tr>
<tr><td>0x97</td><td>0x15</td><td>0x7A</td><td>0x6F</td><td>ID3</td><td>ID2</td><td>ID1</td><td>ID0</td><td>8 бит</td><td>9 байт</td><td>24 бит</td><td>24 бит (младшая часть CRC-32)</td></tr>
</table>

6.2.1 Поле Preamble (Преамбула)

Преамбула служит для обнаружения пакета в эфире. Алгоритм демодуляции, реализованный в базовой станции, использует данное поле для обнаружения пакета в эфире и синхронизации его последующей обработки.

6.2.2 Поле Modem_ID (Идентификатор, присвоенный устройству)

Поле Modem_ID содержит идентификатор, присвоенный устройству. Поле Modem_ID должно иметь размер 32 бита [для номерной емкости сети, составляющей 2^{32} (4 294 967 296) устройств]. Порядок следования байт – от старшего к младшему.

6.2.3 Поле Crypto Iter (Криптоитератор)

Поле Crypto Iter используется для реализации механизма защиты данных. Размер данного поля 8 бит. Формирование поля Crypto Iter (криптоитератора) должно выполняться в соответствии с алгоритмом, описанным в разделе В.2 приложения В.

6.2.4 Поле Payload (Данные транспортного уровня)

Поле Payload должно иметь размер 9 байт. Данное поле должно содержать зашифрованное значение пакета данных транспортного уровня. Поле Payload должно быть зашифровано в соответствии с алгоритмом, описанным в разделе В.3 приложения В.

6.2.5 Поле MIC0_7 (Имитовставка)

Поле MIC0_7 используется для реализации механизма защиты данных. Размер данного поля 24 бит. Формирование поля MIC0_7 должно выполняться в соответствии с алгоритмом, описанным в разделе В.4 приложения В.

6.2.6 Поле Packet CRC (Контрольная сумма пакета данных)

Поле Packet CRC содержит контрольную сумму CRC32 полей Modem_ID, Payload, Crypto Iter и MIC0_7. Используются три младших байта данного значения. Порядок следования байт – от старшего к младшему. Программный код реализации функции вычисления данного параметра (CRC32) на языке Си приведен в разделе Е.5 приложения Е.

6.2.7 Поле Error correction code (Помехоустойчивый код)

Поле Error correction code является кодовым словом помехоустойчивого кода для коррекции ошибок, и должно вычисляться из данных поля Error correction code source.

Поле Error correction code source (входные данные для кодера помехоустойчивого кода) совместно составляют поля Modem_ID, Crypto Iter, Payload, MIC0_7 и Packet CRC.

Могут использоваться два различных помехоустойчивых кода:

- Несистематический сверточный код (255, 363) [1], из которого с помощью метода выкалывания получен код скорости 5/8. Исходные коды кодирования и метода выкалывания приведены в приложении И.1. Базовая станция должна принимать из радиоэфира сообщения, отправленные с использованием данного кода.

- Несистематический полярный код [2] скорости 5/8. При использовании такого кода данные не передаются в канале в явном виде, кодовое слово длиной 32 байта вычисляется на основании поля Error correction code source длиной 20 байт и передается в канал.

Используемые таблицы данных для кодирования, а также исходные коды программной реализации помехоустойчивого кодирования на языке Си приведены в приложении И.2. Поддержка декодирования кода со стороны базовой станции необязательна.

6.3 DOWNLINK-пакет

Общая длина DOWNLINK-пакета должна составлять 36 байт. Размер поля Payload (данные транспортного уровня) должен составлять 9 байт. Программный код реализации функции формирования DOWNLINK-пакета на языке Си приведен в разделе Е.2 приложения Е.

Формат DOWNLINK-пакетов одинаков для различных скоростей передачи данных. Порядок следования битов в байтах DOWNLINK-пакета – от старшего к младшему.

Структура формата DOWNLINK-пакета приведена в таблице 5.

Таблица 5 – Структура формата DOWNLINK-пакета

<table>
<tr><td rowspan="2">Preamble (Преамбула)</td><td rowspan="2">Crypto Iter (Криптоитератор)</td><td rowspan="2">Payload (Данные транспортного уровня)</td><td rowspan="2">MIC0_7 (Имитовставка)</td><td colspan="2">Определение ошибки и коррекция</td></tr>
<tr><td>Packet CRC (Контрольная сумма пакета данных)</td><td>Error correction code (Помехоустойчивый код)</td></tr>
<tr><td>32 бит</td><td>8 бит</td><td>9 байт</td><td>24 бит</td><td>24 бит (младшая часть CRC-32)</td><td>16 байт</td></tr>
<tr><td></td><td colspan="4">Error correction code source
(Входные данные для кодера помехоустойчивого кода)
(16 байт)</td><td></td></tr>
</table>

6.3.1 Поле Preamble (Преамбула)

Преамбула служит для обнаружения пакета в эфире. Значение данного поля должно быть сформировано в соответствии с алгоритмом, приведенным в приложении Г.

6.3.2 Поле Crypto Iter (Криптоитератор)

Поле Crypto Iter используется для реализации механизма защиты данных. Размер данного поля 8 бит. Формирование поля Crypto Iter (криптоитератора) должно выполняться в соответствии с алгоритмом, описанным в разделе В.2 приложения В.

6.3.3 Поле Payload (Данные транспортного уровня)

Поле Payload должно иметь размер 9 байт. Данное поле должно содержать зашифрованное значение пакета данных транспортного уровня. Поле Payload должно быть зашифровано в соответствии с алгоритмом, описанным в разделе В.3 приложения В.

6.3.4 Поле MIC0_7 (Имитовставка)

Поле MIC0_7 используется для реализации механизма защиты данных. Размер данного поля 24 бит. Формирование поля MIC0_7 должно выполняться в соответствии с алгоритмом, описанным в разделе В.4 приложения В.

6.3.5 Поле Packet CRC (Контрольная сумма пакета данных)

Поле Packet CRC содержит контрольную сумму CRC32 полей Modem_ID, Payload, Crypto Iter и MIC0_7. Используются три младших байта данного значения. Порядок следования байт – от старшего к младшему. Программный код реализации функции вычисления данного параметра (CRC32) на языке Си приведен в разделе Е.5 приложения Е.

6.3.6 Поле Error correction code (Помехоустойчивый код)

Поле Error correction code является кодовым словом помехоустойчивого кода для коррекции ошибок, и должно вычисляться из данных поля Error correction code source.

Поле Error correction code source (входные данные для кодера помехоустойчивого кода) совместно составляют поля Crypto Iter, Payload, MIC0_7 и Packet CRC.

Для помехоустойчивого кодирования используется Zigzag код [3].

Используемые таблицы данных для кодирования, а также исходные коды программной реализации помехоустойчивого кодирования на языке Си приведены в приложении Ж.

7 Транспортный уровень (Transport layer)

7.1 Общие положения

Транспортный уровень обеспечивает механизмы приема и передачи данных уровня приложения и управляющих команд между оконечным устройством NB-Fi и сервером NB-Fi (используя базовые станции NB-Fi) либо между двумя оконечными устройствами NB-Fi.

Транспортный уровень описывает следующие функции:

- подтверждения доставки сообщений;
- повторной отправки данных;
- разбиения больших пакетов данных на фрагменты и последующего их «склеивания»;
- буферизации отправки данных;
- синхронизации системного времени;
- конфигурирования режимов работы;
- автоматического выбора режима работы (скорости, мощности передачи);
- автоматического перехода на более предпочтительные рабочие диапазоны частот.

Ключевые особенности транспортного уровня NB-Fi, позволяющие протоколу в наибольшей степени соответствовать задачам построения LPWAN-сетей:

- низкое количество «накладных» данных, используемых для транспортного уровня;

- организация группового квитирования пакетов, позволяющее экономить использование канала связи при подтверждении приема;

- реализация специальных режимов работы для устройств с батарейным питанием.

Для передачи данных от устройства к базовой станции должны использоваться UPLINK-пакеты. Для передачи данных от базовой станции к должны использоваться DOWNLINK-пакеты. Допускается работа в режиме передачи данных от устройства к устройству («режим peer-to-peer»), при этом должны использоваться DOWNLINK-пакета для передачи в обоих направлениях.

Реализация функций транспортного уровня предполагает буферизацию пакетов данных в виде программного стека. Глубина приемного буфера должна быть 32 пакета. Это обусловлено разрядностью поля ITER (итератор), равной 5 бит, которое должно использоваться для циклической нумерации всех пакетов и их последующей идентификации при запросах повторной отправки. Таким образом, передающий узел должен хранить 32 последних отправленных пакета для их возможной переотправки при последующем обмене данными.

7.2 Описание функций транспортного уровня

7.2.1 Режимы работы

Основные режимы работы транспортного уровня NB-Fi и их описание приведены в таблице 6.

Таблица 6 – Основные режимы работы транспортного уровня (параметр NBFI_MODE

Режим работы	Описание
NRX (No RX)	Передача данных от устройства к серверу. Устройство передает данные при необходимости, остальное время модем находится в режиме «сон». Не поддерживается переотправка «потерянных» данных и режим автоматического выбора оптимальной скорости и диапазона частот.
DRX (Discontinuous RX)	Передача данных в обоих направлениях. Устройство передает данные при необходимости и переходит в режим приема на непродолжительное время сразу после окончания передачи. Сервер буферизирует все запросы на отправку данных устройству и выполняет передачу данных во время «открытия» временного «окна», когда устройство переходит в режим приема. Возможна работа в режиме переотправки «потерянных» данных и режиме автоматического выбора скоростей и диапазонов частот. Данный режим используют для устройств с батарейным питанием.

Режим работы	Описание
CRX (Continuous RX)	Передача данных в обоих направлениях. Устройство передает данные при необходимости, в остальное время находится в режиме приема. Отправка данных на устройство с сервера возможна в любой момент. Все функции протокола работают в полном объеме. Возможна передача данных «peer-to-peer». Данный режим используют для устройств со стационарным питанием либо для кратковременного перехода в него с целью обмена «peer-to-peer».
OFF	Прием и передача данных отключены.

Режимы подтверждения доставки данных и их описание приведены в таблице 7.

Таблица 7 – Режимы подтверждения доставки данных (параметр NBFI_HANDSHAKE_MODE)

Режим работы	Описание
HANDSHAKE_NONE	Подтверждение доставки выключено
HANDSHAKE_SIMPLE	Подтверждение доставки включено

Режимы группового подтверждения доставки данных и их описание приведены в таблице 8.

Таблица 8 – Режимы группового подтверждения доставки данных (параметр NBFI_MACK_MODE)

Режим работы	Описание
MACK_0	Подтверждение доставки выключено
MACK_1	Подтверждение каждого пользовательского пакета
MACK_2	Подтверждение каждых двух пользовательских пакетов
MACK_4	Подтверждение каждых четырех пользовательских пакетов
MACK_8	Подтверждение каждых восьми пользовательских пакетов
MACK_16	Подтверждение каждых 16-ти пользовательских пакетов
MACK_32	Подтверждение каждых 32-х пользовательских пакетов

7.2.2 Обеспечение надежности доставки данных в режимах CRX и DRX

Для обеспечения надежности доставки данных должна применяться отправка пакетов в режиме HANDSHAKE_SIMPLE. Отправка должна выполняться посредством сеансов обмена данными между передающим и приемным узлами. При отправке пользовательских пакетов (длина данных равна 8 байт), количество пакетов, входящих в сеанс отправки, должно определяться параметром MACK. Значения MACK более единицы

позволяют уменьшить количество отправляемых пакетов подтверждения приема. При групповой отправке количество пакетов, входящих в сеанс отправки, должно быть равно количеству пакетов в группе. Каждый отправляемый пакет должен содержать в своем заголовке HEADER поле ITER, которое должно инкрементироваться для каждого последующего пакета. Значения поля ITER должно изменяться в диапазоне 0–31. Передающий узел (устройство или сервер), выполняя отправку последнего пакета из группы, должен выставлять в нем флаг ACK в заголовке пакета. Данный флаг сигнализирует приемной стороне, что необходимо отправить в ответ системный пакет ACK_P.

ACK_P должен содержать 32-битную маску, каждый бит которой должен содержать информацию об успешном приеме пакета с номером итератора, соответствующим позиции бита в маске. Формат пакета ACK_P описан в [0].

Передающий узел должен обработать ACK_P пакет и повторно отправить один или несколько пакетов, которые не были доставлены и которые соответствуют данному сеансу отправки.

Если передающий узел не получил ACK_P пакет в ответ на запрос, он должен повторить отправку последнего пакета по истечении заданного интервала времени (тайм-аута) NBFI_RX_TIMEOUT.

При значении параметра WAIT_ACK_TIMEOUT [см. 0, перечисление ш)], не равном 0, NBFI_RX_TIMEOUT = WAIT_ACK_TIMEOUT.

При значении параметра WAIT_ACK_TIMEOUT, равном 0, NBFI_RX_TIMEOUT должен вычисляться по формуле

$$NBFI\_RX\_TIMEOUT = NBFI\_UL\_DELAY + NBFI\_DL\_LISTEN\_TIME + random()\%NBFI\_DL\_ADD\_RND\_LISTEN\_TIME \quad (7.1)$$

Параметры NBFI_UL_DELAY, NBFI_DL_LISTEN_TIME и NBFI_DL_ADD_RND_LISTEN_TIME должны зависеть от выбранных режимов скорости передачи данных. Их значения приведены в таблицах 9 - 11.

Таблица 9 – Значение параметра NBFI_UL_DELAY в зависимости от режима скорости передачи данных

NBFI_TX_PHY_CHANNEL	TIMEOUT, мс
UL_DBPSK_50_PROT_E	5900
UL_DBPSK_400_PROT_E	740
UL_DBPSK_3200_PROT_E	95
UL_DBPSK_25600_PROT_E	15

Таблица 10 – Значение параметра NBFI_DL_LISTEN_TIME в зависимости от режима скорости передачи данных

NBFI_RX_PHY_CHANNEL	TIMEOUT, мс
DL_DBPSK_50_PROT_D	60000
DL_DBPSK_400_PROT_D	30000

DL_DBPSK_3200_PROT_D	6000
DL_DBPSK_25600_PROT_D	6000

Таблица 11 – Значение параметра NBFI_DL_ADD_RND_LISTEN_TIME в зависимости от режима скорости передачи данных

NBFI_RX_PHY_CHANNEL	TIMEOUT, мс
DL_DBPSK_50_PROT_D	5000
DL_DBPSK_400_PROT_D	1000
DL_DBPSK_3200_PROT_D	100
DL_DBPSK_25600_PROT_D	100

Повторные отправки пакетов должны выполняться до тех пор, пока не будет получен ответ либо не будет превышено число повторных отправок, заданное параметром NUM_OF_RETRIES [см. 0, перечисление г)]. Повторная отправка пакетов, вызванная получением ACK_P пакета, также должна входить в расчет общего числа повторов и быть ограничена параметром NUM_OF_RETRIES.

В случае успешной отправки всех пакетов, входящих в сеанс, передающий узел должен отправить системный пакет CLEAR [см. 0] (либо CLEAR_T [см. 0]), информирующий приемный узел, что все данные переданы и необходимо выполнить очистку истории принятых данных.

При неуспешном выполнении сеанса отправки данных от устройства к серверу передающий узел должен выполнить сброс всех параметров работы NB-Fi к значениям по умолчанию и отослать системный пакет SYNC [см. 0], уведомляющий приемный узел о смене режима работы. Должна быть реализована возможность отключения сброса параметров и отправки SYNC-пакета при помощи флага FLG_NO_RESET_TO_DEFAULTS, содержащегося в параметре NBFI_ADDITIONAL_FLAGS.

На рисунке 7.1 приведен пример успешной отправки группового пакета[1]. В рамках данного сеанса выполнена отправка трех пакетов с итераторами 14–16. В ответ на пакет с итератором 16, содержащий флаг ACK, сервер отправил ACK_P пакет с маской, сообщающей, что сообщения с итераторами 14–16 успешно приняты. Приняв данный пакет, передающий узел, отправил CLEAR_T-пакет, означающий успешное завершение сеанса и содержащий информацию у текущем времени устройства.

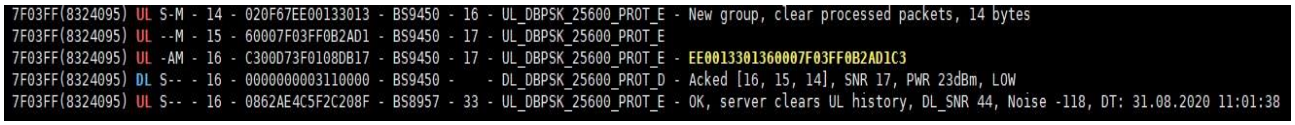

Рисунок 7.1 – Пример успешной отправки группового пакета

[1] *Взят из log-файла обмена на телеком-сервере WAVIoT компании ООО «Телематические Решения»*

На рисунке 7.2 приведен пример успешной отправки группового пакета с повторной отправкой пакетов. Групповой пакет, состоящий из пакетов с итераторами 26 – 28 был доставлен в результате следующих шагов:

1) Из трех пакетов, составляющих группу, был принят только пакет с итератором 28, который является завершающим и требующим подтверждения. Сервер отправил ACK_P пакет, подтверждающий прием только пакета с итератором 28.
2) Передающий узел, получив ACK_P пакет, отправило повторно пакеты с итераторами 26 и 27, запросив последним пакетом подтверждение доставки.
3) Из двух отправленных пакетов сервер принял только последний пакет с итератором 27 и отправил в ответ ACK_P пакет, подтверждающий прием пакетов с итераторами 27 и 28.
4) Передающий узел, получив ACK_P пакет, отправило повторно пакет с итератором 26, запросив подтверждение доставки.
5) Сервер получил пакет с итератором 26, успешно сформировал групповой пакет, и отправил устройству ACK_P пакет, подтверждающий прием данного пакета.
6) Приняв данный пакет, передающий узел, отправил CLEAR_T-пакет, означающий успешное завершение сеанса и содержащий информацию у текущем времени устройства.

```
7F08D1(8325329) UL -AM - 28 - C3003F4001088E17 - BS9394 - 30 - UL_DBPSK_25600_PROT_E
7F08D1(8325329) DL S-- - 28 - 00000000001E0000 - BS9394 -    - DL_DBPSK_25600_PROT_D - Acked [28], SNR 30, PWR 20dBm, LOW
7F08D1(8325329) UL -AM - 27 - 60007F08D10C17D1 - BS9394 - 30 - UL_DBPSK_25600_PROT_E
7F08D1(8325329) DL S-- - 27 - 00400000001E0000 - BS9394 -    - DL_DBPSK_25600_PROT_D - Acked [27, 28], SNR 30, PWR 20dBm, LOW
7F08D1(8325329) UL SAM - 26 - 020F8DEE00133013 - BS9394 - 33 - UL_DBPSK_25600_PROT_E - EE0133013600007F08D10C17D1C3
7F08D1(8325329) DL S-- - 26 - 0000000000210000 - BS9394 -    - DL_DBPSK_25600_PROT_D - Acked [26], SNR 33, PWR 20dBm, LOW
7F08D1(8325329) UL S-- - 26 - 08BCB24C5F19208C - BS9394 - 30 - UL_DBPSK_25600_PROT_E - OK, server clears UL history, DL_SNR 25, Noise -118, DT: 31.08.2020 11:20:12
```

Рисунок 7.2– Пример успешной отправки группового пакета с повторной отправкой пакетов

[1] Взят из log-файла обмена на телеком-сервере WAVIoT компании ООО «Телематические Решения»

В режиме работы HANDSHAKE_NONE передающий узел не выполняет запрос ACK_P пакетов, и передача данных в этом случае выполняется без контроля успешной доставки (верификации).

7.2.3 Передача групповых пакетов

Пакеты данных большой длины (не умещающиеся в одном пакете MAC-уровня) должны быть отправлены при помощи группы пакетов. Максимальное число пакетов в группе должно быть равно 30. Суммарная максимальная длина группового пакета должна составлять 240 байт.

Возможность групповой отправки данных должна быть реализована как в режимах работы с верификацией доставки данных, так и без нее. Первый пакет в группе должен являться системным пакетом типа GROUP, и содержать информацию об общей длине группы, а также контрольной сумме данных, содержащихся в группе. Размер контрольной суммы должен составлять 1 байт. Каждый пакет группы должен содержать флаг MULTI в своем заголовке. На приемной стороне должна выполняться обработка входных пакетов и «склеивание» пакетов в один блок данных.

Формат пакета GROUP приведен в [0].

Программный код реализации функции вычисления контрольной суммы CRC8 на языке Си приведен в разделе Е.3 приложения Е.

7.2.4 Передача коротких пакетов (длиной менее 8 байт)

Пакеты, имеющие длину, менее чем 8 байт, должны отправляться при помощи системных SHORT пакетов. Длина полезных данных должна быть указана в младших 7 битах первого байта данных SHORT пакета. Данный параметр должен иметь значения от 7 до 127. Формат SHORT пакета приведен в [0].

7.2.5 Разрешение коллизий одновременной передачи

Базовые станции стандарта NB-Fi должны выполнять одновременный прием множества каналов. Это позволяет не учитывать при реализации транспортного уровня коллизии, вызванные одновременной передачей пакетов различными устройствами.

Так, как оконечные устройства NB-Fi работают в режиме «полудуплекс», выполняя единовременно либо прием, либо передачу данных, возможно возникновение коллизий в тот момент, когда базовая станция и устройство выполняют встречную передачу пакетов.

Для минимизации вероятности возникновения данных коллизий поведение устройств и сервера должно определяться с учетом следующих правил:

1) При отправке очередного пакета данных передающая сторона должна всегда выставлять в заголовке пакета флаг MULTI в том случае, если вслед за данным пакетом планируется отправка следующего. Приемная сторона, принимая пакет с флагом MULTI, должна продолжать выполнять прием и не выполнять передачу до получения пакета без флага MULTI либо до истечения заданного интервала времени (тайм-аута) NBFI_DL_LISTEN_TIME. Величина данного тайм-аута должна различаться для разных скоростей приема и соответствовать значениям, приведенным в таблице 10.

2) При повторных отправках пакетов по причине неполучения ACK_P пакета в ответ на запрос флага ACK используемый тайм-аут должен иметь различные значения от повтора к повтору. Это позволяет избежать циклического повторения коллизии из-за постоянства значений тайм-аутов. Переменная составляющая тайм-аута повторов определяется параметром NBFI_DL_ADD_RND_LISTEN_TIME. Величина данного тайм-аута должна различаться для разных скоростей приема и соответствовать случайному числу в диапазоне от 0 до значения, приведенного в таблице 11.

7.2.6 Автоматический выбор оптимальной скорости передачи данных

Устройства, работающие в режимах DRX и CRX, должны выполнять автоматический контроль качества радиосигнала (как приема, так и передачи) и производить смену скоростей передачи данных, опираясь на средние значения соотношений сигнал/шум (SNR) для UPLINK и DOWNLINK-пакетов.

Расчет SNR на входе приемника устройства должен выполняться при приеме пакетов, используя данные, предоставляемые аппаратурой радиотрансивера, входящего в

состав устройства. Расчет SNR передаваемого сигнала должен выполняться на основании данных, которые измеряет базовая станция на входе своего приемника и которые передаются устройству в одном из полей ACK_P пакета.

Решения о смене скорости передачи данных UPLINK-пакетов либо DOWNLINK-пакетов должно принимать оконечное устройство, анализируя уровни SNR.

При достаточно высоком значении SNR (≥SNRLEVEL_FOR_UP + TXSNRDEG либо ≥SNRLEVEL_FOR_UP + RXSNRDEG) устройство должно выполнять перевод скорости на один уровень выше. Так как базовая станция принимает одновременно все поддерживаемые скорости передачи UPLINK-пакетов, смена скоростей UPLINK должна осуществляться без уведомления сервера. При смене скорости DOWNLINK-пакетов устройство должно выполнять отправку системного SYNC пакета, содержащего новые параметры скоростного режима. Данный пакет должен формироваться с флагом ACK и требовать подтверждение приема. В ответ на данный системный пакет сервер должен отправлять SACK_P-пакет [см. 0]. Прием пакета SACK_P устройство должно выполнять уже на новой скорости. В отсутствие ответного пакета от сервера устройство должно выполнять повторные отправки SYNC пакетов. Если после определенного числа повторных попыток доставки SYNC пакета, равного NUM_OF_RETRIES, пакет SACK_P так и не был получен, устройство должно возвращать свой скоростной режим в предыдущее состояние, отсылая SYNC пакет, содержащий предыдущие значения параметров скорости. В этот раз SYNC пакет должен отсылаться единожды без запроса подтверждения.

Повышение скорости UPLINK либо DOWNLINK должно выполняться при условии, что базовая станция поддерживает более высокие скоростные режимы. Об этом сервер должен уведомлять устройство, выставляя флаги UL_SPEED_NOT_MAX и DL_SPEED_NOT_MAX, содержащиеся в пакете SACK_P.

Устройство должно повышать скорость передачи данных до тех пор, пока уровень SNR не снизится до значения, не позволяющего выполнять дальнейшее повышение, либо пока не будет достигнуто максимальное значение скорости.

Если при достижении максимального уровня скорости передачи UPLINK-пакетов уровень SNR имеет достаточное значение (≥SNRLEVEL_FOR_UP + TXSNRDEG), устройство должно выполнять постепенное (с шагом 3 дБ) снижение мощности передатчика.

Если при достижении максимального уровня скорости приема DOWNLINK-пакетов уровень SNR имеет достаточное значение (≥SNRLEVEL_FOR_UP + RXSNRDEG), устройство должно уведомить сервер о необходимости снижения выходной мощности передатчика базовой станции, выставляя флаг DL_POWER_STEP_DOWN в пакете ACK_P, отправляемом от устройства к серверу.

Если уровень SNR снижается ниже заданного значения (SNRLEVEL_FOR_DOWN), устройство должно выполнять обратные действия. Сначала должна повышаться выходная мощность передатчиков устройства или базовой станции, затем выполняться ступенчатое снижение скорости передачи либо приема данных.

Для уведомления сервера о необходимости повышения выходной мощности должен использоваться флаг DL_POWER_STEP_UP в пакете ACK_P, отправляемом от устройства к серверу.

Пороговые уровни для повышения и понижения скоростей либо мощности и поправочные коэффициенты для различных скоростей приведены в таблицах 12-14.

Таблица 12 – Пороговые уровни для повышения и понижения скоростей либо мощности

Параметр	Значение порога, дБ
SNRLEVEL_FOR_UP	15
SNRLEVEL_FOR_DOWN	10

Таблица 13 – Значение поправочного коэффициента TXSNRDEG для различных скоростей передачи

NBFI_TX_PHY_CHANNEL	Поправочное значение, дБ
UL_DBPSK_50_PROT_E	0
UL_DBPSK_400_PROT_E	9
UL_DBPSK_3200_PROT_E	18
UL_DBPSK_25600_PROT_E	27

Таблица 14 – Значение поправочного коэффициента RXSNRDEG для различных скоростей приема

NBFI_RX_PHY_CHANNEL	Поправочное значение, дБ
DL_DBPSK_50_PROT_D	0
DL_DBPSK_400_PROT_D	9
DL_DBPSK_3200_PROT_D	18
DL_DBPSK_25600_PROT_D	27

Механизм автоматического выбора скоростного режима работы устройства может быть отключен при помощи флага FLG_FIXED_BAUD_RATE, содержащегося в параметре NBFI_ADDITIONAL_FLAGS.

Механизм автоматического управления мощностью передачи может быть отключен при помощи флага FLG_NO_REDUCE_TX_PWR, содержащегося в параметре NBFI_ADDITIONAL_FLAGS.

Сервер, отправляя пакет SACK_P, имеет возможность сообщить устройству о необходимости изменения параметра FPLAN, определяющего частотную сетку каналов приема и передачи. Алгоритм формирования частоты приема и передачи для оконечного устройства описан в приложении А. Изменяя значение параметра FPLAN, сервер может переключить устройство на работу в другой, более предпочтительной полосе прима и/или передачи.

Пакет SACK_P, также содержит значение идентификатора базовой станции, через которую осуществляется обмен данными с сервером, либо значение идентификатора

сервера, с которым осуществляется обмен. В случае если сервер выполняет изменение параметра FPLAN, SACK_P пакет должен содержать идентификатор базовой станции, в противном случае – идентификатор сервера.

На рисунке 7.3 приведен пример сеанса обмена пакетами, выполняющий повышение скоростей передачи и приема[2].

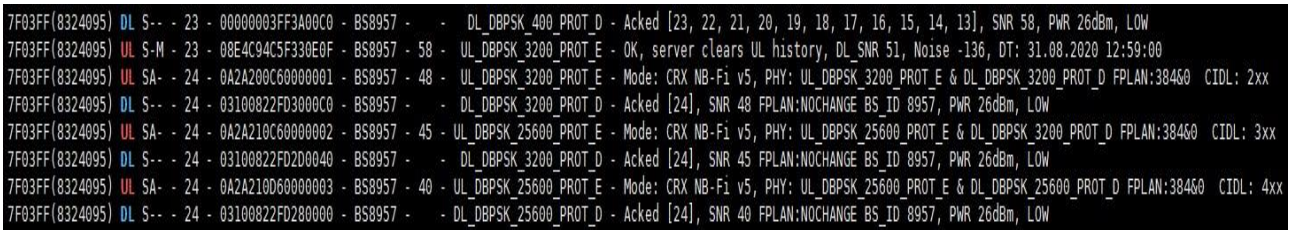

Рисунок 7.3 – Пример сеанса обмена пакетами, выполняющий повышение скоростей передачи и приема

7.2.7 Работа в режиме DRX

Данный режим предназначен для устройств, которые должны обладать очень малыми значениями среднего потребления энергии (например, устройства с батарейным питанием). Проблема реализации радиосвязи в таких устройствах заключается в невозможности выполнения приема данных в непрерывном режиме. DRX – это режим, который специально разработан с целью решить данную проблему максимально эффективно.

Отличие от режима CRX состоит в том, что устройство должно кратковременно переходить в режим приема сразу после окончания передачи последнего пакета из группы пакетов, отправляемых непрерывно друг за другом. Сервер должен выполнять отправку данных только во время действия данного «временного окна».

Таким образом, сеанс обмена данными должен происходить по инициативе устройства. Сервер, при необходимости, должен инициировать дополнительные сеансы обмена, используя флаг MULTI в заголовке последнего пакета, отправляемого устройству в рамках предыдущего сеанса.

Сеанс отправки данных от устройства к серверу должен выполняться в полном объеме до подтверждения приема всех пакетов. Следующий за ним сеанс отправки данных от сервера к устройству может быть прерван из-за потерь данных при плохом уровне сигнала и затем должен быть продолжен при следующем «выходе устройства на связь».

7.2.8 Синхронизация системного времени

При обмене данными в режиме подтверждения доставки должен быть реализован механизм синхронизации времени между сервером и оконечным устройством. Сценарий взаимодействия между сервером и устройством следующий:

- оконечное устройство, выполнив отправку данных и получив пакеты, подтверждающие их доставку, должно отправить системный пакет CLEAR_T,

---

[2] Взят из log-файла обмена на телеком-сервере WAVIoT компании ООО «Телематические Решения»

информирующий приемный узел (сервер) о том, что все данные переданы и можно выполнить очистку истории принятых данных. Помимо этого, в данном пакете должно содержаться значение текущего (на момент отправки пакета) времени системных часов передающего узла (устройства). Время должно передаваться в виде 4-байтного значения, соответствующего формату времени Unix Timestamp (количество секунд от 1 января 1970 года) и часовому поясу UTC. Формат пакета CLEAR_T описан в [0].

- сервер должен сравнить полученное время с собственным системным временем и в том случае, если разница превышает значение 8191 с, отослать системный пакет SENDTIME, содержащий значение текущего времени в формате Unix Timestamp. Пакет отсылается без запроса подтверждения. Формат пакета SENDTIME описан в [0].

Если разница не превышает значение 8191 с, но более 5 с, то сервер должен сохранить это значение и при следующем сеансе обмена выполнить коррекцию системного времени устройства, отправив величину данной поправки внутри пакета ACK_P. Формат пакета ACK_P описан в [0].

Таким образом, при регулярной отправке данных в режимах работы CRX и DRX должна выполняться синхронизация времени оконечного устройства с точностью до 5 секунд. Преимуществом данного механизма является практически полное отсутствие отправки дополнительных пакетов в обоих направлениях.

7.2.9 Конфигурирование параметров NB-Fi

Режимы функционирования протокола NB-Fi конкретного устройства должны определяться совокупностью параметров NB-Fi.

Все параметры NB-Fi устройства должны быть доступны для чтения и записи при помощи системных CONF-пакетов. Формат данных пакетов и наименование параметров описаны в [0].

7.2.10 Идентификация типа устройства

Для идентификации типа устройства должны быть использованы три идентификатора, размером 16 бит:

- идентификатор производителя – параметр, уникальный для каждого производителя устройств;
- идентификатор устройства – параметр, уникальный для каждого типа устройств;
- идентификатор протокола – параметр, однозначно определяющий формат данных и логику работы устройства на уровне приложения.

Данные идентификаторы сервер должен запрашивать у устройства, при их отсутствии в базе данных сервера либо устройство должно отправлять их самостоятельно при первом включении, периодически или при изменении значений.

Отправка идентификаторов типа устройства должна осуществляться при помощи CONF-пакета с полем данных APP_IDS [см. 0, перечисление x)].

Сервер NB-Fi должен использовать данные идентификаторы для передачи их вместе со всеми принятыми от устройства ULAPP-пакетами на уровень приложения, где в зависимости от значения идентификаторов будет определяться, поддерживается ли данное

устройство и каким образом необходимо выполнять взаимодействие с данным устройством.

7.2.11 Работа в режиме peer-to-peer

Данный режим предполагает реализацию обмена данными напрямую между двумя оконечными устройствами, без участия базовых станций и сервера NB-Fi.

В режиме peer-to-peer для передачи данных в обоих направлениях должны быть использованы DOWNLINK-пакеты.

Параметры скоростей приема и передачи, мощности передачи и значение параметра FPLAN должны быть неизменны на протяжении всей работы в режиме peer-to-peer.

7.3 Описание пакетов данных транспортного уровня

7.3.1 Формат пакета транспортного уровня

Структура формата пакета транспортного уровня приведена в таблице 15.

Таблица 15 – Структура формата пакета транспортного уровня

HEADER (Заголовок)				DATA (Данные)
SYS	ACK	MULTI	ITER	
1 бит	1 бит	1 бит	5 бит	8 байт

Описание полей:

- SYS – флаг системного пакета;
- ACK – флаг, информирующий о том, что данный пакет требует подтверждения;
- MULTI:

1) флаг групповой посылки,

2) флаг, информирующий приемную сторону о том, что непосредственно вслед за данным пакетом будет отправлен следующий;

- ITER – итератор пакета;
- DATA – поле данных пакета;

Пакеты транспортного уровня разделяются на:

- пользовательские пакеты;
- системные пакеты.

Системные пакеты используют для передачи служебной информации, для реализации механизмов транспортного протокола, а также для передачи полезной информации (при передаче пакетов типа GROUP и SHORT).

Данные уровня приложения, длина которых равна 8 байт, передаются внутри пользовательского пакета транспортного уровня.

Данные уровня приложения длиной более 8 байт передаются путем дробления на пакеты транспортного уровня и объединения их в групповую посылку. При этом, первый пакет в группе является системным (GROUP), а остальные – пользовательскими.

Данные уровня приложения, длина которых меньше 8 байт, передаются внутри системного (SHORT) пакета.

Структура формата пользовательского пакета приведена в таблице 16, структура формата системного пакета – в таблице 17.

Таблица 16 – Структура формата пользовательского пакета

HEADER (Заголовок)				DATA (Данные)
SYS (1 бит)	ACK (1 бит)	MULTI (1 бит)	ITER (5 бит)	PAYLOAD
0	0/1	0/1	От 0 до 31	8 байт

Таблица 17 – Структура формата системного пакета

HEADER (Заголовок)				DATA (Данные)	
SYS	ACK	MULTI	ITER	TYPE	SYS_PAYLOAD
1	0/1	0/1	От 0 до 31	1 байт	7 байт

Описание полей:

- SYS – флаг системного пакета;
- ACK – флаг, информирующий о том, что данный пакет требует подтверждения;
- MULTI:

1) флаг групповой посылки,

2) флаг, информирующий приемную сторону о том, что за данным пакетом будет отправлен другой;

- ITER – итератор пакета;
- DATA – поле данных пакета;
- TYPE – тип системного пакета;
- PAYLOAD – полезные данные пользовательского пакета;
- SYS_PAYLOAD – полезные данные системного пакета.

7.3.2 Типы системных пакетов

Основные типы системных пакетов, используемые в протоколе NB-Fi, приведены в таблице 18.

Таблица 18 – Основные типы системных пакетов

CODE (код)	TYPE (тип)	Описание
0b1xxxxxxx	SHORT (см. 0)	Короткий пакет данных
0x00	ACK_P (см. 0)	Подтверждение приема пакетов
0x01	HEARTBEAT (см. 0)	Информация о параметрах работы устройства
0x02	GROUP (см. 0)	Первый пакет в группе
0x03	SACK_P (см. 0)	Подтверждение приема системного пакета
0x04	CLEAR (см. 0)	Сигнал завершения сеанса передачи данных
0x06	CONF (см. 0)	Настройка параметров NB-Fi
0x07	RESET (см. 0)	Сброс устройства
0x08	CLEAR_T (см. 0)	Сигнал завершения сеанса передачи данных со значением Unix Timestamp
0x09	SENDTIME (см. 0)	Сигнал синхронизации системного времени
0x0A	SYNC (см. 0)	Информация о состоянии ключевых параметров NB-Fi

7.3.2.1 Пакет SHORT (короткий пакет данных)

Данный системный пакет предназначен для передачи пользовательских данных длиной менее 8 байт.

Структура формата пакета SHORT приведена в таблице 19.

Таблица 19 – Структура формата пакета SHORT

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0b1xxxxxxx	SHORT	0	0x80 + LENGTH
		1 - 7	PAYLOAD

7.3.2.2 Пакет ACK_P (подтверждение приема пакетов)

Данный системный пакет предназначен для подтверждения приема одного или нескольких пакетов данных.

Структура формата пакета ACK_P приведена в таблице 20.

Таблица 20 – Структура формата пакета ACK_P

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x00	ACK_P	0	0x00
		1–4	MASK [см. 0, а)]
		5	SNR [см. 0, б)]
		6	NOISE_OR_RTC_OFS_0_7 [см. 0, в)]
		7	MFLAGS [см. 0, г)]

а) MASK

Структура формата поля MASK приведена в таблице 21.

Таблица 21 – Структура формата поля MASK

MASK0	MASK1	MASK2	MASK3
bit 0: статус пакета с итератором (i −32) – bit 7: статус пакета с итератором (i −25)	bit 0: статус пакета с итератором (i −24) – bit 7: статус пакета с итератором (i −17)	bit 0: статус пакета с итератором (i −16) – bit 7: статус пакета с итератором (i − 9)	bit 0: статус пакета с итератором (i − 8) – bit 7: статус пакета с итератором (i − 1)

Примечание – i-итератор ACK_P пакета равен итератору принятого пакета, в ответ на который отправлен ACK_P.

Статус сообщения с текущим итератором i не включен в маску, так как факт получения пакета ACK_P подтверждает успешность доставки пакета с текущим итератором.

б) SNR (Соотношение сигнал/шум пакета)

Соотношение сигнал/шум пакета, в ответ на который отправлен ACK_P, используют для оценки качества связи при автоматическом выборе скорости и мощности передатчика. Допустимые значения – от 0 до 127 дБ.

в) NOISE_OR_RTC_OFS_0_7 (Уровень входного шума либо 8 младших бит поправки времени)

В данном поле передается уровень входного шума NOISE либо 8 младших бит поправки времени RTC_OFS_0_7. Первый вариант используется при передаче от устройства к серверу в качестве дополнительной информации; второй вариант – при передаче от сервера к устройству в случае применения механизма коррекции времени. Допустимые значения – от 0 до 255. Данные значения соответствуют уровням шума от минус 150 до плюс 105 дБм либо 8 младшим битам 14-битной поправки времени. Значение поля, равное 0 означает отсутствие необходимости коррекции времени.

г) MFLAGS

Структура формата поля MFLAGS при передаче ACK_P от сервера к устройству приведена в таблице 22.

Таблица 22 – Структура формата поля MFLAGS при передаче ACK_P от сервера к устройству

7 бит	6 бит	От 5 до 0 бит
UL_SPEED_NOT_MAX	DL_SPEED_NOT_MAX	RTC_OFS_8_13

UL_SPEED_NOT_MAX уведомляет устройство о том, что текущая скорость передачи UPLINK-пакетов является не максимальной для данной базовой станции. Активное значение – 1.

DL_SPEED_NOT_MAX уведомляет устройство о том, что текущая скорость приема DOWNLINK-пакетов является не максимальной для данной базовой станции. Активное значение – 1.

RTC_OFS_8_13 – старшие 6 бит 14-битной поправки времени – используют при реализации механизма коррекции системного времени оконечного устройства.

Структура формата поля MFLAGS при передаче ACK_P от устройства к серверу приведена в таблице 23.

Таблица 23 – Структура формата поля MFLAGS при передаче ACK_P от устройства к серверу

7 бит	6 бит	От 5 до 0 бит
DL_POWER_STEP_DOWN	DL_POWER_STEP_UP	TX_PWR

DL_POWER_STEP_DOWN уведомляет сервер о необходимости снизить мощность передатчика базовой станции для данного устройства. Активное значение – 1.

DL_POWER_STEP_UP уведомляет сервер о необходимости повысить мощность передатчика базовой станции для данного устройства. Активное значение – 1.

TX_PWR соответствует уровню текущей мощности передатчика устройства. Допустимые значения – от 0 до RF_MAX_POWER дБм.

7.3.2.3 Пакет HEARTBEAT (информация о параметрах работы устройства)

Данный системный пакет предназначен для передачи информации о параметрах работы устройства.

Структура формата пакета HEARTBEAT приведена в таблице 24.

Таблица 24 – Структура формата пакета HEARTBEAT

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x01	HEARTBEAT	0	0x01
		1	0x00
		2	VSUP [см. 0, а)]
		3	TEMP [см. 0, б)]
		4	AVER_RX_SNR [см. 0, в)]
		5	AVER_TX_SNR [см. 0, г)]
		6	NOISE [см. 0, д)]
		7	TX_PWR [см. 0, е)]

а) VSUP (Напряжение питания устройства)

Должно быть представлено в следующем формате:
Старший бит: 1, если $U \geq 3$В.
Младшие 7 бит: десятые доли вольта.
Формула для перевода данного поля в вольты следующая:

V = 2 + (D>>7) + (D&0x7F)/100

б) TEMP (Температура внутри устройства)

Тип данных: int8_t.
Допустимые значения: от минус 128 °С до плюс 128 °С.

в) AVER_RX_SNR (Среднее значение соотношения сигнал/шум на входе приемника устройства)

Среднее значение соотношения сигнал/шум на входе приемника устройства определяют по нескольким последним принятым пакетам. Допустимые значения: от 0 до 127 дБ.

г) AVER_TX_SNR (Среднее значение соотношения сигнал/шум на входе приемника базовой станции)

Среднее значение соотношения сигнал/шум на входе приемника базовой станции определяют по нескольким последним принятым пакетам и вычисляют на основании данных, получаемых в поле SNR ACK_P пакета. Допустимые значения: от 0 до 127 дБ.

д) NOISE (Уровень шума на входе приемника устройства)

Допустимые значения: от 0 до 255. Данные значения соответствуют уровням шума от минус 150 до плюс 105 дБм.

е) TX_PWR (Уровень текущей мощности передатчика устройства)

Допустимые значения: от минус 10 до RF_MAX_POWER дБм.

7.3.2.4 Пакет GROUP (первый пакет в группе)

Данный системный пакет предназначен для определения заголовка (первого пакета) групповой посылки.

Структура формата пакета GROUP приведена в таблице 25.

Таблица 25 – Структура формата пакета GROUP

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x02	GROUP	0	0x02
		1	GROUP_LEN [см. 0, а)]
		2	GROUP_CRC [см. 0, б)]
		3–7	PAYLOAD_0_4 [см. 0, в)]

а) GROUP_LEN

Суммарное количество байт полезных данных во всей групповой посылке.

б) GROUP_CRC

Контрольная сумма CRC8 данных группы.

в) PAYLOAD_0_4

Первые 5 байт полезных данных групповой посылки.

7.3.2.5 Пакет SACK_P (подтверждение приема системного пакета)

Данный системный пакет предназначен для подтверждения приема сервером системного пакета данных, отправленного оконечным устройством.

Структура формата пакета SACK_P приведена в таблице 26.

Таблица 26 – Структура формата пакета SACK_P

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x03	SACK_P	0	0x03
		1–2	SET_FPLAN [см. 0, а)]
		3-4	BS_OR_SERVER_ID [см. 0, б)]
		5	SNR [см.0, б)]
		6	NOISE_OR_RTC_OFS_0_7 [см. 0, в)]
		7	MFLAGS [см. 0, г)]

а) SET_FPLAN

Данное поле может содержать новое значение параметра FPLAN либо код, означающий, что параметр FPLAN должен остаться без изменений.

Параметр FPLAN определяет характеристики рабочей полосы частот, используемой для передачи и приема данных. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

Значение поля SET_FPLAN, равное **4104** означает, что параметр FPLAN должен остаться без изменений.

Структура формата параметра FPLAN приведена в таблице 27.

Таблица 27 – Структура формата параметра FPLAN

BIT # (номер бита)	FPLAN
0–2 (LSB)	DL_OFFSET
3	DL_SIGN
4-5	DL_WIDTH
6-11	UL_OFFSET
12	UL_SIGN
13-15(MSB)	UL_WIDTH

Влияние параметров, приведенных в таблице 27, на выбор частот приема и передачи описано приложении А.

б) BS_OR_SERVER_ID

Данное поле предназначено для передачи оконечному устройству значения идентификатора базовой станции BS_ID, через которую осуществляется обмен данными, либо идентификатора сервера SERVER_ID, с которым выполняется взаимодействие. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

При SET_FPLAN = **4104** параметр BS_OR_SERVER_ID содержит идентификатор BS_ID.
При SET_FPLAN ≠ **4104** параметр BS_OR_SERVER_ID содержит идентификатор SERVER_ID

7.3.2.6 Пакет CLEAR (Сигнал завершения сеанса передачи данных)

Данный системный пакет предназначен для информирования о завершении сеанса передачи данных.

Структура формата пакета CLEAR приведена в таблице 28.

Таблица 28 – Структура формата пакета CLEAR

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x04	CLEAR	0	0x04
		1–7	Зарезервированы

7.3.2.7 Пакет CONF (настройка параметров NB-Fi)

Данный системный пакет предназначен для выполнения настройки параметров NB-Fi.

Структура формата пакета CONF приведена в таблице 29.

Таблица 29 – Структура формата пакета CONF

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Поле данных)
0x06	CONF	0	0x06
		1	CMD&PARAM [см. 0, а)]
		2–7	CONF_DATA

Поле CONF_DATA определено в пунктах 0, перечисления г)–ш).

а) CMD&PARAM

Структура формата поля CMD&PARAM приведена в таблице 30.

Таблица 30 – Структура формата поля CMD&PARAM

От 7 до 6 бит	От 5 до 0 бит
CMD (см. 0, б)	PARAM [см. 0, в)]

б) CMD

Структура формата поля CMD приведена в таблице 31.

Таблица 31 – Структура формата поля CMD

CODE (код)	TYPE (тип)	Описание
0x00	READ_CMD	Команда чтения параметра(ов)
0x01	WRITE_CMD	Команда записи параметра(ов)
0x03	WRITE_CMD_SAVE	Команда записи параметра(ов) с сохранением в энергонезависимой памяти

в) PARAM

Структура формата поля PARAM приведена в таблице 32.

Таблица 32 – Структура формата поля PARAM

CODE (код)	TYPE (тип)	Read/Write (чтение/ запись)	Наименование параметра(ов)
0x00	NBFI_PARAM_MODE	R/W	Параметры режима работы NB-Fi

CODE (код)	TYPE (тип)	Read/Write (чтение/ запись)	Наименование параметра(ов)
0x01	NBFI_PARAM_HANDSHAKE	R/W	Параметры подтверждения доставки данных
0x03	NBFI_PARAM_TXFREQ	R/W	Параметр TXFREQ (частота передачи)
0x04	NBFI_PARAM_RXFREQ	R/W	Параметр RXFREQ (частота приема)
0x05	NBFI_PARAM_ANT	R/W	Параметры RF-фронтенда (выбор антенн и мощности передачи)
0x07	NBFI_PARAM_HEART_ BEAT	R/W	Параметры отправки HEARTBEAT-пакетов
0x08	NBFI_PARAM_TX_ BRATES	R	Чтение поддерживаемых скоростей передачи
0x09	NBFI_PARAM_RX_ BRATES	R	Чтение поддерживаемых скоростей приема
0x0A	NBFI_PARAM_ VERSION	R	Чтение версии исполнения устройства
0x0B	NBFI_ADD_FLAGS	R/W	Параметр ADDITIONAL_FLAGS
0x0C	NBFI_QUALITY	R	Чтение параметра QUALITY
0x0D	NBFI_UL_BASE_FREQ	R/W	Параметр UL_BASE_FREQ (базовая частота передачи)
0x0E	NBFI_DL_BASE_FREQ	R/W	Параметр DL_BASE_FREQ (базовая частота приема)
0x0F	NBFI_QUALITY_EX	R	Чтение параметра QUALITY_EXT
0x11	APP_IDS	R	Чтение идентификаторов типа устройства
0x12	BSANDSERVER_IDS	R	Чтение идентификаторов базовой станции и сервера
0x13	FPLAN	R/W	Параметр FPLAN (частотный план)
0x14	WAIT_ACK_TIMEOUT	R/W	Параметр WAIT_ACK_TIMEOUT (время ожидания подтверждения доставки)

Описание поля CONF_DATA для каждого параметра таблицы 32 приведено в пунктах 0, перечисления г) – ш).

г) Значение поля CONF_DATA для параметра NBFI_PARAM_MODE (PARAM = 0x00)

Таблица 33 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_MODE

BYTE # (номер байта)	CONF_DATA
0	NBFI_MODE
1	NBFI_MACK_MODE
2	NBFI_TX_PHY_CHANNEL
3	NBFI_RX_PHY_CHANNEL
4	NBFI_TX_PWR
5	NBFI_NUM_OF_RETRIES

NBFI_MODE – режим работы транспортного уровня протокола NB-Fi. Описание поля NBFI_MODE приведено в таблице 34.

Таблица 34 – Описание поля NBFI_MODE

CODE (код)	TYPE (тип)
0	NRX
1	DRX
2	CRX
4	OFF

NBFI_MACK_MODE – параметр группового подтверждения доставки данных (multiple ack) при отправке. Может иметь значения от 0 до 32.

NBFI_TX_PHY_CHANNEL – параметр, определяющий тип и скорость пакетов при отправке UPLINK-пакетов, используемый в текущий момент времени. Описание поля NBFI_TX_PHY_CHANNEL приведено в таблице 35.

Таблица 35 – Описание поля NBFI_TX_PHY_CHANNEL

CODE (код)	TYPE (тип)	Описание
30	UL_DBPSK_50_PROT_E	UPLINK-пакет на скорости 50 бит/с
31	UL_DBPSK_400_PROT_E	UPLINK-пакет на скорости 400 бит/с

CODE (код)	TYPE (тип)	Описание
32	UL_DBPSK_3200_PROT_E	UPLINK-пакет на скорости 3200 бит/с
33	UL_DBPSK_25600_PROT_E	UPLINK-пакет на скорости 25600 бит/с
21	UL_DBPSK_50_PROT_D	DOWNLINK-пакет на скорости 50 бит/с
24	UL_DBPSK_400_PROT_D	DOWNLINK-пакет на скорости 400 бит/с
26	UL_DBPSK_3200_PROT_D	DOWNLINK-пакет на скорости 3200 бит/с
28	UL_DBPSK_25600_PROT_D	DOWNLINK-пакет на скорости 25600 бит/с

Примечание – DOWNLINK-пакеты применяют для отправки при взаимодействии «peer-to-peer».

NBFI_RX_PHY_CHANNEL – параметр, определяющий тип и скорость пакетов при приеме DOWNLINK-пакетов, используемый в текущий момент времени. Описание поля NBFI_RX_PHY_CHANNEL приведено в таблице 36.

Таблица 36 – Описание поля NBFI_RX_PHY_CHANNEL

CODE (код)	TYPE (тип)	Описание
10	DL_DBPSK_50_PROT_D	DOWNLINK-пакет на скорости 50 бит/с
11	DL_DBPSK_400_PROT_D	DOWNLINK-пакет на скорости 400 бит/с
12	DL_DBPSK_3200_PROT_D	DOWNLINK-пакет на скорости 3200 бит/с
13	DL_DBPSK_25600_PROT_D	DOWNLINK-пакет на скорости 25600 бит/с

NBFI_TX_PWR – уровень выходной мощности передатчика. Тип параметра: int8_t. Допустимые значения: от минус 10 до RF_MAX_POWER дБм.

NBFI_NUM_OF_RETRIES – максимальное количество повторных отправок пакетов в течение одного сеанса отправки данных. Допустимые значения: от 0 до 255.

д) Значение поля CONF_DATA для параметра NBFI_PARAM_HANDSHAKE (PARAM = 0x01)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_HANDSHAKE приведено в таблице 37.

Таблица 37 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_HANDSHAKE

BYTE # (номер байта)	CONF_DATA
0	NBFI_HANDSHAKE_MODE
1	NBFI_MACK_MODE
2–5	Зарезервированы

NBFI_HANDSHAKE_MODE – режим подтверждения доставки. Описание поля NBFI_HANDSHAKE_MODE приведено в таблице 38.

Таблица 38 – Описание поля NBFI_HANDSHAKE_MODE

CODE (код)	TYPE (тип)
0	HANDSHAKE_NONE
1	HANDSHAKE_SIMPLE

NBFI_MACK_MODE – параметр группового подтверждения доставки данных (multiple ack) при отправке. Допустимые значения: от 0 до 32.

е) Значение поля CONF_DATA для параметра NBFI_PARAM_TXFREQ (PARAM = 0x03)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_TXFREQ приведено в таблице 39.

Таблица 39 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_TXFREQ

BYTE # (номер байта)	CONF_DATA
0–3	TXFREQ
4–5	Зарезервированы

TXFREQ – параметр, определяющий частоту отправки данных. При TXFREQ = 0 частоту отправки данных вычисляют в соответствии с алгоритмом, приведенным разделе А.1 приложения А.

При TXFREQ = 0 чтение данного параметра возвращает значение 0 и не позволяет определить текущую частоту отправки.

При TXFREQ ≠0 частота отправки равна значению TXFREQ.

Порядок следования байт в данном поле данных – старшим байтом вперед.

ж) Значение поля CONF_DATA для параметра NBFI_PARAM_RXFREQ (PARAM = 0x04)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_RXFREQ приведено в таблице 40.

Таблица 40 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_RXFREQ

BYTE # (номер байта)	CONF_DATA
0–3	RXFREQ
4–5	Зарезервированы

RXFREQ – параметр, определяющий частоту, на которой выполняется прием данных. При RXFREQ = 0 частоту f вычисляют в соответствии с алгоритмом, приведенным разделе А.2 приложения А.

При RXFREQ = 0 чтение данного параметра возвращает значение 0 и не позволяет определить текущую частоту приема.

При RXFREQ ≠ 0 частота приема равна значению RXFREQ.

Порядок следования байтов в данном поле данных – старшим байтом вперед.

и) Значение поля CONF_DATA для параметра NBFI_PARAM_ANT (PARAM = 0x05)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_ANT приведено в таблице 41.

Таблица 41 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_ANT

BYTE # (номер байта)	CONF_DATA
0	NBFI_TX_PWR
1	NBFI_TX_ANT
2	NBFI_RX_ANT
3–5	Зарезервированы

NBFI_TX_PWR – уровень выходной мощности передатчика. Тип параметра: int8_t. Допустимые значения: от минус 10 до RF_MAX_POWER дБм.

NBFI_TX_ANT – параметр, определяющий, какой из RF-трактов используется при передаче. Значения параметра и соответствующие им варианты RF-трактов зависят от конкретной реализации устройства.

NBFI_RX_ANT – параметр, определяющий, какой из RF-трактов используется при приеме. Значения параметра и соответствующие им варианты RF-трактов зависят от конкретной реализации устройства.

к) Значение поля CONF_DATA для параметра NBFI_PARAM_HEART_BEAT (PARAM = 0x07)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_HEART_BEAT приведено в таблице 42.

Таблица 42 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_HEART_BEAT

BYTE # (номер байта)	CONF_DATA
0	NBFI_HEARTBEAT_NUM
1–2	NBFI_HEARTBEAT_INTERVAL
3–5	Зарезервированы

NBFI_HEARTBEAT_NUM – параметр, определяющий количество HEARTBEAT-пакетов, которые будут отправлены после сброса устройства. При NBFI_HEARTBEAT_NUM = 255 отправка HEARTBEAT-пакетов выполняется неограниченное количество раз.

NBFI_HEARTBEAT_INTERVAL – параметр, определяющий периодичность отправки HEARTBEAT-пакетов. Допустимые значения: от 0 до 65535. Порядок следования байтов в поле параметра: старшим байтом вперед. Порядок расчета периодичности отправки HEARTBEAT-пакетов при разных режимах работы транспортного уровня протокола NB-Fi (NBFI_MODE) приведен в таблице 43.

Таблица 43 – Порядок расчета периодичности отправки HEARTBEAT-пакетов при разных режимах работы транспортного уровня протокола NB-Fi

NBFI_MODE	Периодичность отправки HEARTBEAT-пакетов, с
NRX, DRX	NBFI_HEARTBEAT_INTERVAL * 60
CRX	NBFI_HEARTBEAT_INTERVAL

м) Значение поля CONF_DATA для параметра NBFI_PARAM_TX_BRATES (PARAM = 0x08)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_TX_BRATES приведено в таблице 44.

Таблица 44 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_TX_BRATES

BYTE # (номер байта)	CONF_DATA
0–5	NBFI_TX_BRATES

NBFI_TX_BRATES – в данном поле перечислены все поддерживаемые устройством скорости передачи данных, которые используются при автоматическом выборе оптимальной скорости. Данные представлены в формате значений параметра NBFI_TX_PHY_CHANNEL [см. таблицу 35]. Максимально возможное количество поддерживаемых скоростей равно шести. Поддерживаемые скорости перечислены начиная с младшего байта данного поля. Если число скоростей менее шести, старшие (неиспользуемые) байты данного поля равны 0xFF.

Примечание – Параметр NBFI_PARAM_TX_BRATES доступен только для чтения.

н) Значение поля CONF_DATA для параметра NBFI_PARAM_RX_BRATES (PARAM = 0x09)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_RX_BRATES приведено в таблице 45.

Таблица 45 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_RX_BRATES

BYTE # (номер байта)	CONF_DATA
0–5	NBFI_RX_BRATES

NBFI_RX_BRATES – в данном поле перечислены все поддерживаемые устройством скорости приема данных, которые используются при автоматическом выборе оптимальной скорости. Данные представлены в формате значений параметра NBFI_RX_PHY_CHANNEL [см. таблицу 36]. Максимально возможное количество поддерживаемых скоростей равно шести. Поддерживаемые скорости перечислены начиная с младшего байта данного поля. Если число скоростей менее шести, старшие (неиспользуемые) байты данного поля равны 0xFF.

Примечание – Параметр NBFI_PARAM_RX_BRATES доступен только для чтения.

п) Значение поля CONF_DATA для параметра NBFI_PARAM_VERSION (PARAM = 0x0A)

Описание структуры формата поля CONF_DATA для параметра NBFI_PARAM_VERSION приведено в таблице 46.

Таблица 46 – Структура формата поля CONF_DATA для параметра NBFI_PARAM_VERSION

BYTE # (номер байта)	CONF_DATA
0	NBFI_REV
1	NBFI_SUBREV
2	COMP_CRC
3	HARDWARE_ID
4	HARDWARE_REV
5	BAND_ID

NBFI_REV – версия транспортного протокола NB-Fi. Текущее значение равно 5.

NBFI_SUBREV – дополнительная версия транспортного протокола NB-Fi – в данный момент не используется.

COMP_CRC – контрольная сумма CRC8 даты и времени компиляции программного обеспечения устройства – используется для автоматического контроля модификаций программного обеспечения.

HARDWARE_ID – идентификатор аппаратной платформы устройства.

HARDWARE_REV – версия аппаратной платформы устройства.

BAND_ID – идентификатор радиочастотного решения устройства – описывает рабочие частоты приемного и передающего трактов.

Примечание – Параметр NBFI_PARAM_VERSION доступен только для чтения.

р) Значение поля CONF_DATA для параметра NBFI_ADD_FLAGS (PARAM = 0x0B)

Описание структуры формата поля CONF_DATA для параметра NBFI_ADD_FLAGS приведено в таблице 47.

Таблица 47 – Структура формата поля CONF_DATA для параметра NBFI_ADD_FLAGS

BYTE # (номер байта)	CONF_DATA
0-1	NBFI_ADDITIONAL_FLAGS
4–5	Зарезервированы

NBFI_ADDITIONAL_FLAGS – параметр, представляющий собой совокупность дополнительных флагов, управляющих определенными функциями протокола, сгруппированных в виде битовой маски. Активное значение каждого флага – 1.

Порядок следования байтов в поле параметра: младшим байтом вперед.

Описание поля NBFI_ADDITIONAL_FLAGS приведено в таблице 48.

Таблица 48 – Описание поля NBFI_ADDITIONAL_FLAGS

BIT # (номер бита)	FLAG (дополнительные флаги)
0 (LSB)	FLG_FIXED_BAUD_RATE
1	FLG_NO_RESET_TO_DEFAULTS
2	FLG_NO_SENDINFO
3	FLG_SEND_ALOHA
4	Зарезервирован
5	FLG_LBT
6	FLG_OFF_MODE_ON_INIT
7	FLG_DO_NOT_SEND_PKTS_ON_START
8	FLG_SHORT_RANGE_CRYPTO
9-15 (MSB)	Зарезервированы

FLG_FIXED_BAUD_RATE – данный флаг деактивирует механизм автоматического выбора оптимальной скорости и мощности передачи.

FLG_NO_RESET_TO_DEFAULTS – данный флаг деактивирует функцию сброса режима скоростей к значениям по умолчанию после неудачного выполнения сеанса передачи данных и используется совместно с флагами FLG_FIXED_BAUD_RATE и FLG_SHORT_RANGE_CRYPTO при соединениях «peer-to-peer».

FLG_NO_SEND_INFO – данный флаг отключает автоматическую отправку информационных пакетов.

FLG_SEND_ALOHA– данный флаг включает режим отправки данных «ALOHA», при котором каждый пакет MAC-уровня отсылается дважды на различных частотах с целью снижения вероятности потери пакета в результате коллизии.

FLG_LBT – данный флаг активирует режим передачи LBT (Listen Before Talk).

FLG_OFF_MODE_ON_INIT – данный флаг сигнализирует устройству переходить в режим работы OFF при инициализации устройства.

FLG_DO_NOT_SEND_PKTS_ON_START – данный флаг отключает отправку системных пакетов CLEAR и CONF при инициализации устройства.

FLG_SHORT_RANGE_CRYPTO – данный флаг включает режим защиты данных без смены ключей. Описание данного режима приведено в разделе В.2 приложения В. Используется совместно с флагами FLG_FIXED_BAUD_RATE и FLG_NO_RESET_TO_DEFAULTS при соединениях «peer-to-peer».

с) Значение поля CONF_DATA для параметра NBFI_QUALITY (PARAM = 0x0C)

Описание структуры формата поля CONF_DATA для параметра NBFI_QUALITY приведено в таблице 49.

Таблица 49 – Структура формата поля CONF_DATA для параметра NBFI_QUALITY

BYTE # (номер байта)	CONF_DATA
0–1	UL_TOTAL
2–3	DL_TOTAL
4	AVER_RX_SNR
5	AVER_TX_SNR

UL_TOTAL – общее количество отправленных пакетов с момента сброса устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

DL_TOTAL – общее количество принятых пакетов с момента сброса устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

AVER_RX_SNR – среднее значение соотношения сигнал/шум на входе приемника устройства, определяемое по нескольким последним принятым пакетам. Допустимые значения: от 0 до 127 дБ.

AVER_TX_SNR – среднее значение соотношения сигнал/шум на входе приемника базовой станции, определяемое по нескольким последним принятым пакетам, вычисляется на основании данных, получаемых в поле SNR ACK_P пакета. Допустимые значения: от 0 до 127 дБ.

Примечания

1 Параметр NBFI_QUALITY доступен только для чтения.

2 Параметры DL_TOTAL, AVER_RX_SNR, AVER_TX_SNR актуальны только для режимов работы DRX и CRX с NBFI_HANDSHAKE_MODE = HANDSHAKE_SIMPLE.

т) Значение поля CONF_DATA для параметра NBFI_UL_BASE_FREQ (PARAM = 0x0D)

Описание структуры формата поля CONF_DATA для параметра NBFI_UL_BASE_FREQ приведено в таблице 50.

Таблица 50 – Структура формата поля CONF_DATA для параметра NBFI_UL_BASE_FREQ

BYTE # (номер байта)	CONF_DATA
0–3	UL_BASE_FREQ
4–5	Зарезервированы

UL_BASE_FREQ – параметр соответствует базовой частоте, используемой при расчете частоты, на которой отправляются данные.

у) Значение поля CONF_DATA для параметра NBFI_DL_BASE_FREQ (PARAM = 0x0E)

Описание структуры формата поля CONF_DATA для параметра NBFI_DL_BASE_FREQ приведено в таблице 51.

Таблица 51 – Структура формата поля CONF_DATA для параметра NBFI_DL_BASE_FREQ

BYTE # (номер байта)	CONF_DATA
0–3	DL_BASE_FREQ
4–5	Зарезервированы

DL_BASE_FREQ – параметр соответствует базовой частоте, используемой при расчете частоты, на которой принимаются данные.

ф) Значение поля CONF_DATA для параметра NBFI_QUALITY_EX (PARAM = 0x0F)

Описание структуры формата поля CONF_DATA для параметра NBFI_QUALITY_EX приведено в таблице 52.

Таблица 52 – Структура формата поля CONF_DATA для параметра NBFI_QUALITY_EX

BYTE # (номер байта)	CONF_DATA
0	UL_RATING
1	DL_RATING
2–3	UL_SUCCESS_TOTAL
4–5	UL_FAULT_TOTAL

UL_RATING – показатель уровня сигнала на входе приемника – вычисляется на основании текущего режима скорости и среднего значения соотношения сигнал/шум на входе приемника. Допустимые значения: от 0 до 10.

DL_RATING – показатель уровня сигнала от данного устройства на входе приемника базовой станции – вычисляется на основании текущего режима скорости и среднего значения соотношения сигнал/шум на входе базовой станции. Допустимые значения: от 0 до 10.

UL_SUCCESS_TOTAL – общее количество успешно доставленных пакетов с момента сброса устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед. Для режима работы NRX либо в случае NBFI_HANDSHAKE_MODE = HANDSHAKE_NONE данный параметр равен общему количеству отправленных пакетов.

UL_FAULT_TOTAL – общее количество недоставленных пакетов с момента сброса устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед. Пакет считается недоставленным, если его не удалось доставить в результате сеанса передачи данных.

Примечания

1 Параметр NBFI_QUALITY_EX доступен только для чтения.

2 Параметры UL_RATING, DL_RATING, UL_FAULT_TOTAL актуальны только для режимов работы DRX и CRX с NBFI_HANDSHAKE_MODE = HANDSHAKE_SIMPLE.

x) Значение поля CONF_DATA для параметра APP_IDS (PARAM = 0x11)

Описание структуры формата поля CONF_DATA для параметра APP_IDS приведено в таблице 53.

Таблица 53 – Структура формата поля CONF_DATA для параметра APP_IDS

BYTE # (номер байта)	CONF_DATA
0–1	MANUFACTURER_ID
2–3	HARDWARE_TYPE_ID
4-5	PROTOCOL_ID

MANUFACTURER_ID – идентификатор производителя устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

HARDWARE_TYPE_ID – идентификатор типа устройства. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

PROTOCOL_ID – идентификатор протокола уровня приложения. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

Примечание – Параметр APP_IDS доступен только для чтения.

ц) Значение поля CONF_DATA для параметра BSANDSERVER_IDS (PARAM = 0x12)

Описание структуры формата поля CONF_DATA для параметра BSANDSERVER_IDS приведено в таблице 54.

Таблица 54 – Структура формата поля CONF_DATA для параметра BSANDSERVER_IDS

BYTE # (номер байта)	CONF_DATA
0–2	BS_ID
3–5	SERVER_ID

BS_ID – идентификатор базовой станции через которую осуществляется обмен данными с сервером NB-Fi. Тип данного параметра: uint24_t. Порядок следования байтов в данном поле: старшим байтом вперед.

SERVER_ID – идентификатор сервера NB-Fi, с которым осуществляется обмен данными. Тип данного параметра: uint24_t. Порядок следования байтов в данном поле: старшим байтом вперед.

Примечание – Параметр BSANDSERVER_IDS доступен только для чтения.

ч) Значение поля CONF_DATA для параметра FPLAN (PARAM = 0x13)

Описание структуры формата поля CONF_DATA для параметра FPLAN приведено в таблице 55.

Таблица 55 – Структура формата поля CONF_DATA для параметра FPLAN

BYTE # (номер байта)	CONF_DATA
0–1	FPLAN
2–5	Зарезервированы

FPLAN – данный параметр определяет текущее значение рабочей полосы частот, используемой для передачи и приема данных. Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

ш) Значение поля CONF_DATA для параметра WAIT_ACK_TIMEOUT (PARAM = 0x14)

Описание структуры формата поля CONF_DATA для параметра WAIT_ACK_TIMEOUT приведено в таблице 56.

Таблица 56 – Структура формата поля CONF_DATA для параметра WAIT_ACK_TIMEOUT

BYTE # (номер байта)	CONF_DATA
0–1	WAIT_ACK_TIMEOUT
2–5	Зарезервированы

WAIT_ACK_TIMEOUT – параметр, определяющий время ожидания приема ACK_P пакета, подтверждающего получение данных.

Тип данного параметра: uint16_t. Порядок следования байтов в данном поле: старшим байтом вперед.

При WAIT_ACK_TIMEOUT = 0 время ожидания определяется в соответствии с формулой (7.1) [см. 0].

При WAIT_ACK_TIMEOUT ≠ 0 время ожидания равно значению WAIT_ACK_TIMEOUT в миллисекундах.

7.3.2.8 Пакет RESET (сброс устройства)

Данный системный пакет предназначен для выполнения процедуры программного перезапуска устройства.

Структура формата пакета RESET приведена в таблице 57.

Таблица 57 – Структура формата пакета RESET

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x07	RESET	0	0x07
		1	0xDE
		2	0xAD
		3–7	Зарезервированы

7.3.2.9 Пакет CLEAR_T (Сигнал завершения сеанса передачи данных со значением Unix Timestamp)

Структура формата пакета CLEAR_T приведена в таблице 58.

Таблица 58 – Структура формата пакета CLEAR_T

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x08	CLEAR_T	0	0x08

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
		1–4	UTS
		5	SNR [см. 0, б)]
		6	NOISE_OR_RTC_OFS_0_7 [см. 0, в)]
		7	MFLAGS [см. 0, г)]

UTS – 4-байтное значение системного времени в формате Unix Timestamp (количество секунд от 1 января 1970 г.), соответствующее часовому поясу UTC.

7.3.2.10 Пакет SENDTIME (Сигнал синхронизации системного времени)

Структура формата пакета SENDTIME приведена в таблице 59.

Таблица 59 – Структура формата пакета SENDTIME

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x09	SENDTIME	0	0x09
		1–4	UTS
		5–7	Зарезервированы

UTS – 4-байтное значение системного времени в формате Unix Timestamp (количество секунд от 1 января 1970 г.), соответствующее часовому поясу UTC.

7.3.2.11 Пакет SYNC (Информация о состоянии ключевых параметров NB-Fi)

Пакет данного типа используется оконечным устройством для уведомления сервера об изменении состояния параметров NB-Fi, информация о которых необходима серверу для корректной отправки данных оконечному устройству. Также данный пакет отсылается оконечным устройством каждый раз при потере связи с сервером.

Использование пакета SYNC при взаимодействии оконечного устройства и сервера описан в [0].

Структура формата пакета SYNC приведена в таблице 60.

Таблица 60 – Структура формата пакета S Y N C

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
0x0A	SYNC	0	0x0A
		1	NBFI_MODE_AND_REV

CODE (код)	TYPE (тип)	BYTE # (номер байта)	DATA (Данные)
		2	NBFI_TX_PHY_CHANNEL [см. таблицу 35]
		3	NBFI_RX_PHY_CHANNEL [см. таблицу 36]
		4-5	FPLAN [см. 7.3.2.5, а)]
		6	CRYPTO_ITER_23_16
		7	CRYPTO_ITER_15_8

Структура формата поля NBFI_MODE_AND_REV приведена в таблице 61.

Таблица 61 – Структура формата поля NBFI_MODE_AND_REV

BIT # (номер бита)	NBFI_MODE_AND_REV
0–2 (LSB)	NBFI_MODE [см. 7.3.2.7, г)]
4-7	NBFI_REV [см. 7.3.2.7, п)]

CRYPTO_ITER_23_16 – значение части криптоитератора передачи от сервера к оконечному устройству от 23-го бита до 16-го бита.

CRYPTO_ITER_15_8 – значение части криптоитератора передачи от сервера к оконечному устройству от 15-го бита до 8-го бита.

Приложение А
(обязательное)
Описание алгоритмов выбора частот приема и передачи

А.1 Алгоритм определения частоты для отправки UPLINK-пакета

Частота отправки UPLINK-пакета зависит от следующих параметров:

NBFI_UL_BASE_FREQ – базовая частота передачи UPLINK-пакетов [см. 0, т)];

UL_WIDTH – ширина рабочей полосы передачи UPLINK-пакетов [см. 0];

UL_OFFSET – смещение рабочей полосы передачи UPLINK-пакетов, относительно базовой частоты [см. 0];

UL_SIGN – направление смещения рабочей полосы передачи UPLINK-пакетов, относительно базовой частоты [см. 0];

Bitrate – скорость передачи UPLINK-пакетов, бит/сек – параметр, определяемый значением параметра NBFI_TX_PHY_CHANNEL [см. таблицу 35];

Modem_ID – идентификатор устройства [см. 0];

MIC0_7 – младшие 8 бит имитовставки [см. 0];

Parity – четность передаваемого пакета, имеет значения 0 или 1 и меняет свое значение с каждым последующим пакетом.

Значения частот в приведенных ниже формулах определяется в Гц.

Ширина рабочей полосы частот, в пределах которой должна осуществляться передача UPLINK-пакетов определяется по формуле

$$\mathrm{ULBandwidth} = 6400 * 2^{\mathrm{DL\_WIDTH}} \quad (A.1)$$

Смещение рабочей полосы частот, относительно базовой частоты определяется по формуле

$$\mathrm{ULBandOffset} = \mathrm{ULBandwidth} * \mathrm{UL\_OFFSET} * (-1) * \mathrm{UL\_SIGN} \quad (A.2)$$

Полоса перестройки несущей частоты передачи UPLINK-пакетов, в зависимости от скорости передачи определяется по формулам

$$\mathrm{ULGap} = \frac{\mathrm{ULBandwidth} - \mathrm{Bitrate} * 2 - 2000}{2}, \text{если } \mathrm{ULBandwidth} > \mathrm{Bitrate} * 2 - 2000, \quad (A.3)$$

$$\mathrm{ULGap} = 0, \text{если } \mathrm{ULBandwidth} \leq \mathrm{Bitrate} * 2 - 2000 \quad (A.4)$$

Смещение несущей частоты заданного пакета, в зависимости от содержания пакета определяется по формуле

$$\mathrm{ULChannelOffset} = \frac{((\mathrm{Modem\_ID} + \mathrm{MIC0\_7}) \bmod 256) * \mathrm{ULGap}}{255} \quad (A.5)$$

Частота для отправки UPLINK-пакета определяется по формулам

$$\text{ULfreq} = \text{NBFI\_UL\_BASE\_FREQ} + \text{ULBandOffset} + \text{ULChannelOffset}, \text{ если parity} = 1, \quad \text{(A.6)}$$

$$\text{ULfreq} = \text{NBFI\_UL\_BASE\_FREQ} + \text{ULBandOffset} - \text{ULChannelOffset}, \text{ если parity} = 0 \quad \text{(A.7)}$$

A.2 Алгоритм определения частоты для отправки DOWNLINK-пакета

Частота отправки DOWNLINK-пакета зависит от следующих параметров:

NBFI_DL_BASE_FREQ – базовая частота передачи DOWNLINK-пакетов [см. 0, y)];

DL_WIDTH – ширина рабочей полосы передачи DOWNLINK-пакетов [см. 0];

DL_OFFSET – смещение рабочей полосы передачи DOWNLINK-пакетов, относительно базовой частоты [см. 0];

DL_SIGN – направление смещения рабочей полосы передачи DOWNLINK-пакетов, относительно базовой частоты [см. 0];

Bitrate – скорость передачи DOWNLINK-пакетов, бит/с – параметр, определяемый значением параметра NBFI_RX_PHY_CHANNEL [см. таблицу 36];

Modem_ID – идентификатор устройства [см. 0].

Значения частот в приведенных ниже формулах определяется в Гц.

Ширина рабочей полосы частот, в пределах которой должна осуществляться передача DOWNLINK-пакетов определяется по формуле

$$\text{DLBandwidth} = 102400 * 2^{\text{DL\_WIDTH}} \quad \text{(A.8)}$$

Смещение рабочей полосы частот, относительно базовой частоты определяется по формуле

$$\text{DLBandOffset} = \text{DLBandwidth} * \text{DL\_OFFSET} * (-1) * \text{DL\_SIGN} \quad \text{(A.9)}$$

Полоса перестройки несущей частоты передачи DOWNLINK-пакетов, в зависимости от скорости передачи определяется по формулам

$$\text{DLGap} = \frac{\text{DLBandwidth} - \text{Bitrate} * 2 - 2000}{2}, \text{если DLBandwidth} > \text{Bitrate} * 2 - 2000, \quad \text{(A.10)}$$

$$\text{DLGap} = 0, \text{если DLBandwidth} \leq \text{Bitrate} * 2 - 2000, \quad \text{(A.11)}$$

Смещение несущей частоты заданного пакета, в зависимости от содержания пакета определяется по формуле

$$\text{DLChannelOffset} = \frac{(\text{Modem\_ID mod } 256) * \text{DLGap}}{255} \quad \text{(A.12)}$$

Частота для отправки DOWNLINK-пакета определяется по формулам

$$\text{DLfreq} = \text{NBFI\_DL\_BASE\_FREQ} + \text{DLBandOffset} + \text{DLChannelOffset}, \text{ если Modem\_ID mod } 2 = 1, \quad \text{(A.13)}$$

$$\text{DLfreq} = \text{NBFI\_DL\_BASE\_FREQ} + \text{DLBandOffset} - \text{DLChannelOffset}, \text{ если Modem\_ID mod } 2 = 0 \quad \text{(A.14)}$$

Приложение Б
(справочное)
Описание алгоритма компенсации нестабильности частот задающего генератора

Суть алгоритма компенсации частот заключается в следующем:

Отправка UPLINK-пакетов устройствами осуществляется на известной (предполагаемой) частоте f_{exp_ul}, вычисляемой по формулам (А.6) и (А.7) приложения А.

Отправка DOWNLINK-пакетов базовой станцией осуществляется на известной (предполагаемой) частоте f_{exp_dl}, вычисляемой по формулам (А.13) и (А.14) приложения А.

Базовая станция, принимая пакет, вычисляет частоту приема f_{rx1} с разрешением в несколько десятков герц (для скорости 50 бит/с) и затем выполняет вычисление обобщенной ошибки частоты Δf_1 по формуле

$$\Delta f_1 = f_{exp\_ul} - f_{rx1}. \quad (Б.1)$$

В состав данной ошибки входят как погрешность передатчика оконечного устройства Δf_{modTX}, так и ошибка измерения частоты базовой станцией Δf_{bsRX}, таким образом,

$$\Delta f_1 = \Delta f_{modTX} + \Delta f_{bsRX}. \quad (Б.2)$$

Данные погрешности имеют достаточно стабильные значения, вызванные начальной неточностью генераторов, и при использовании термокомпенсированных осцилляторов незначительно изменяются при колебаниях температуры.

Передатчик базовой станции, используемый для отправки DOWNLINK-пакетов, также с определенной периодичностью (например, каждые 5 мин) отправляет UPLINK-пакеты на фиксированной частоте f_{test_dl}, которая попадает в полосу частот приема базовой станции. Базовая станция, принимая данный пакет, вычисляет частоту приема f_{rx2} и затем значение обобщенной ошибки частоты Δf_2 по формуле

$$\Delta f_2 = f_{test\_dl} - f_{rx2}. \quad (Б.3)$$

В состав данной ошибки входят как погрешность передатчика базовой станции Δf_{bsTX}, так и ошибка измерения частоты базовой станцией Δf_{bsRX}, таким образом,

$$\Delta f_2 = \Delta f_{bsTX} + \Delta f_{bsRX}. \quad (Б.4)$$

При отправке DOWNLINK-пакетов каждый раз вносится поправка в частоту передачи базовой станции прибавлением обобщенной ошибки Δf_1 и вычитанием обобщенной ошибки Δf_2 по формуле

$$f_{tx} = f_{exp\_dl} + \Delta f_1 - \Delta f_2, \quad \text{(Б.5)}$$

Подставляя в формулу (Б.5) выражения (Б.2) и (Б.3), а также добавляя ошибку передатчика базовой станции Δf_{bsTX}, вычисляют фактическое значение частоты отправки f_{factTX} по формуле

$$\begin{aligned} f_{factTX} &= f_{exp\_dl} + \Delta f_{modTX} + \Delta f_{bsRX} - \Delta f_{bsTX} - \Delta f_{bsRX} + \Delta f_{bsTX} \\ &= f_{exp\_dl} + \Delta f_{modTX}. \end{aligned} \quad \text{(Б.6)}$$

Для успешного приема сообщения данная частота f_{factTX} должна соответствовать фактической частоте приема модема f_{factRX}, которая равна:

$$\Delta f_{factRX} \quad = f_{exp\_dl} + \Delta f_{modRX} \quad \text{(Б.7)}$$

Когда частота приема и частота передачи модема совпадают либо имеют близкие значения, ошибки приема и передачи также приблизительно равны:

$$\Delta f_{modTX} \approx \Delta f_{modRX}. \quad \text{(Б.8)}$$

Таким образом, по формуле (Б.5) вычисляют необходимую компенсацию ошибок генераторов для случая

$$f_{base\_ul} \approx f_{base\_dl} \quad \text{(Б.9)}$$

В общем случае, формула компенсации погрешностей частот имеет следующий вид:

$$f_{tx} = f_{exp\_dl} + \frac{f_{base\_dl}}{f_{base\_ul}}(\Delta f_1 - \Delta f_2). \quad \text{(Б.10)}$$

Подставив в формулу (Б.10) формулы (Б.1) и (Б.3), получаем полную формулу расчета частоты отправки DOWNLINK-пакетов, при которой выполняется коррекция ошибок всех генераторов:

$$f_{tx} = f_{exp\_dl} + \frac{f_{base\_dl}}{f_{base\_ul}}\left(f_{exp\_ul} - f_{rx1} - f_{test\_dl} + f_{rx2}\right). \quad \text{(Б.11)}$$

Приложение В
(обязательное)
Защита данных в протоколе NB-Fi

В.1 Общие положения

Защита данных NB-Fi должна быть реализована на MAC-уровне.

Для защиты данных должно выполняться шифрование блока данных транспортного уровня NB-Fi.

Шифрование данных должно выполняться для UPLINK-пакетов и для DOWNLINK-пакетов.

Для шифрования данных должен применяться алгоритм блочного шифрования «Магма».

Должны быть использованы уникальные для каждого оконечного устройства корневые ключи длиной 256 бит.

Присвоение корневых ключей устройствам должно выполняться при производстве оконечных устройств. Данные ключи должны быть «зашиты» в оконечные устройства и сохранены в базе данных сервера NB-Fi.

Для шифрования UPLINK- и DOWNLINK- пакетов должна использоваться схема формирования ключей шифрования, генерируемых на основании корневого ключа, выделенного устройству. Алгоритм формирования данных ключей приведен в разделе В.2 данного приложения.

Шифрование блока данных транспортного уровня, имеющего размер 9 байт, должно выполняться в соответствии с алгоритмом, описанным приведен в разделе В.3 данного приложения.

Проверка «валидности» принятого пакета должна осуществляться при помощи контроля имитовставки. Алгоритм вычисления имитовставки приведен в разделе В.4 данного приложения.

Для защиты от атаки повторного воспроизведения каждый последующий UPLINK- и DOWNLINK- пакет должен содержать уникальное значение криптоитератора, используемого для формирования ключей и для шифрования данных. Алгоритмы использования значения криптоитератора описаны в разделах В.2 - В.3 данного приложения.

В.2 Алгоритм формирования ключей для шифрования UPLINK- и DOWNLINK-пакетов

Алгоритм формирования ключей должен быть основан на режиме гаммирования симметричного блочного шифрования «Магма» согласно ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015.

Должна применяться схема ключей шифрования, состоящая из 6 ключей, по 3 ключа для UPLINK- и DOWNLINK- пакетов. В каждом комплекте ключей должен находиться мастер-ключ, предназначенный для формирования двух следующих ключей:

- рабочего ключа, с помощью которого происходит шифрования/расшифрование данных;

- ключа имитозащиты, с помощью которого выполняется выработка имитовставки пакета.

Первая пара мастер-ключей должна формироваться из корневого ключа, выделенного устройству.

Каждый комплект ключей должен действовать для обработки 256-ти отправленных пакетов (т.е. для одного периода оборота криптоитератора). При оборачивании криптоитератора должна происходить смена ключей – из мастер-ключа должен формироваться новый мастер-ключ с последующим формированием рабочих ключей и ключей имитозащиты.

Для работы в режиме peer-to-peer необходимо использовать конфигурационный флаг транспортного уровня FLG_SHORT_RANGE_CRYPTO [см. 0, перечисление p)]. При наличии данного флага, смену ключей при оборачивании криптоитератора выполнять не нужно. Таким образом, при работе в режиме peer-to-peer, обеспечивается более низкий уровень криптозащиты данных, но упрощает реализацию механизма синхронизации значений криптоитераторов между узлами peer-to-peer соединения.

Вводится понятие полного криптоитератора: полный криптоитератор – это 32-битный счётчик переданных пакетов, хранящийся на устройстве/сервере.

Младшие 8 бит полного криптоитератора должны передаваться с каждым пакетом MAC-уровня (в поле Crypto Iter – криптоитератор).

Шифрование каждого UPLINK- и DOWNLINK- пакета должно выполняться с использованием нового значения криптоитератора, увеличиваемого на единицу для каждого последующего пакета. Формирование криптоитераторов должно выполняться отдельно для UPLINK- и DOWNLINK-пакетов.

Полное значение UPLINK- и DOWNLINK- криптоитераторов должно храниться в памяти оконечных устройств и сервера NB-Fi.

Вследствие того, что в пакете должны передаваться лишь младшие 8 бит полного криптоитератора, при долгом отсутствии связи сервера с устройством может возникать ситуация, когда ключи устройства и/или сервера рассинхронизированы.

Для противодействия подобным ситуациям должен быть реализован дополнительный алгоритм синхронизации:

- в случае отрицательного результата проверки имитовставки на текущем комплекте ключей, необходимо циклически выполнять проверку имитовставки на последующих комплектах ключей до того момента, пока не будет достигнут положительный результат проверки имитовставки. Полученный комплект ключей должен быть принят основным для дальнейшей работы.

Основные обозначения:

K – ключи, длиной 256 бит;
iv – стартовый вектор 32 бит;
p – открытый/закрытый текст;
$\hat{}n$ – количество повторяемых байт;
K_{root} – корневой ключ известный устройству/серверу;
$K_{xx\ master}$ – мастер-ключ;
$K_{xx\ work}$ – рабочий ключ;

$K_{xx\ mac}$ – ключ имитозащиты;
$K_{ul\ xxx}$ – ключ UPLINK-пакетов;
$K_{dl\ xxx}$ – ключ DOWNLINK-пакетов.

Основной примитив, блочный шифр «Магма» в режиме гаммирования:

$$ctr_{magma}(K, iv, p) \quad \text{(В.1)}$$

Порождение первого мастер-ключа:

$$K_{ul\ master} = ctr_{magma}(K_{root}, 0x00\text{^}4, 0x00\text{^}32) \quad \text{(В.2)}$$
$$K_{dl\ master} = ctr_{magma}(K_{root}, 0xFF\text{^}4, 0x00\text{^}32) \quad \text{(В.3)}$$

Смена мастер-ключей:

$$K_{ul\ master} = ctr_{magma}(K_{ul\ master}, 0x0F\text{^}4, 0x00\text{^}32) \quad \text{(В.4)}$$
$$K_{dl\ master} = ctr_{magma}(K_{dl\ master}, 0x0F\text{^}4, 0x00\text{^}32) \quad \text{(В.5)}$$

Формирование ключей шифрования:

$$K_{ul\ work} = ctr_{magma}(K_{ul\ master}, 0xFF\text{^}4, 0x00\text{^}32) \quad \text{(В.6)}$$
$$K_{dl\ work} = ctr_{magma}(K_{dl\ master}, 0xFF\text{^}4, 0x00\text{^}32) \quad \text{(В.7)}$$

Формирование ключей имитозащиты:

$$K_{ul\ mac} = ctr_{magma}(K_{ul\ master}, 0x00\text{^}4, 0x00\text{^}32) \quad \text{(В.8)}$$
$$K_{dl\ mac} = ctr_{magma}(K_{dl\ master}, 0x00\text{^}4, 0x00\text{^}32) \quad \text{(В.9)}$$

В.3 Алгоритм шифрования блока данных транспортного уровня

В качестве алгоритма шифрования должен использоваться алгоритм симметричного блочного шифрования «Магма» в режиме гаммирования согласно ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015.

$$ctr\_magma\ (K, iv, p) \quad \text{(В.11)}$$

Для стартового вектора *iv* должен использоваться полный криптоитератор, что позволяет задать уникальные начальные условия для шифрования каждого пакета.

В.4 Алгоритм вычисления имитовставки

В качестве алгоритма вычисления имитовставки должен использоваться алгоритм симметричного блочного шифрования «Магма» в режиме выработки имитовставки согласно ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015.

Выработка имитовставки должна производиться методом Encrypt-then-mac, то есть, с начала должны шифроваться данные, затем формироваться имитовставка от зашифрованных данных. В пакет должны быть включены три младших байта сгенерированного результата.

Приложение Г
(обязательное)
Алгоритм определения преамбулы DOWNLINK-пакета

Алгоритм формирования преамбулы для DOWNLINK-пакета должен быть основан на итерационной генерации псевдослучайных чисел с идентификатором модема в качестве стартовой позиции генератора. В процессе каждой итерации должен проверяться фактор корреляции полученной преамбулы. В случае удовлетворения выбранного критерия фактора корреляции, формирование преамбулы считается завершенным.

Фрагменты исходных кодов реализации данного алгоритма приведены в Приложении Д.

Приложение Д
(обязательное)
Фрагменты исходных кодов реализации для определения преамбулы DOWNLINK-пакета

```
//preambula.h

#ifndef _PREAMBULA_H_
#define _PREAMBULA_H_

#ifdef __cplusplus
extern "C"
{
#endif

#include <stdint.h>

#define RAND_MULL       0x1234
#define RAND_ADD        0x10

#define MAX_ITER        100
#define MAX_COEFF       6

uint32_t preambula(uint32_t seed, uint32_t *iter, uint32_t *coeff);

#ifdef __cplusplus
}
#endif

#endif


//preambula.c

#include "preambula.h"
#include <stdint.h>
#include <stdlib.h>

static uint32_t _alu(uint32_t data, uint32_t tmp)
{
        uint32_t test = data ^ tmp;
        int32_t count = 0;
        while (test)
                count += test & 0x01, test >>= 1;
        count = abs(count - (32 - count)) >> 1;
```

```
	return count;
}

static uint32_t _factor(uint32_t data)
{
	uint32_t i, count, tmp, max = 0;
	for (i = 0, tmp = data; i < 32; i++)
	{
		count = _alu(data, tmp);
		tmp <<= 1;
		if (count > max && i)
			max = count;
	}
	for (i = 0, tmp = data; i < 32; i++)
	{
		count = _alu(data, tmp);
		tmp >>= 1;
		if (count > max && i)
			max = count;
	}
	return max;
}

static uint32_t _random(uint32_t seed)
{
	static uint32_t _seed;
	if (!seed)
	{
		_seed = _seed * RAND_MULL + RAND_ADD;
		_seed = _seed << 7 | _seed >> 23;
	}
	else
		_seed = seed;

	return _seed;
}

uint32_t preambula(uint32_t seed, uint32_t *iter, uint32_t *coeff)
{
	uint32_t _rand, _coeff, _iter = 0;
	_random(seed);
	while (_iter < MAX_ITER)
	{
		_rand = _random(0);
		_coeff = _factor(_rand);
```

```
                _iter++;
                if (_coeff < MAX_COEFF)
                        break;
        }
        if (iter)
                *iter = _iter;
        if (coeff)
                *coeff = _coeff;

        return _rand;
}
```

Приложение Е
(справочное)
Фрагменты исходных кодов реализации MAC-уровня

Е.1 Функция формирования и отправки UPLINK-пакета

```
nbfi_status_t NBFi_MAC_TX_ProtocolE(nbfi_transport_packet_t* pkt)
{
 const uint8_t protE_preambula[] = {0x97, 0x15, 0x7A, 0x6F};
 uint8_t ul_buf[20];
 uint8_t ul_buf_encoded[36];
 uint8_t len = 0;
 static _Bool parity = 0;
 uint32_t tx_freq;
 uint32_t mic_or_crc32;

 ul_buf[len++] = Modem_ID[3];
 ul_buf[len++] = Modem_ID[2];
 ul_buf[len++] = Modem_ID[1];
 ul_buf[len++] = Modem_ID[0];

 ul_buf[len++] = nbfi_iter.ul;
 ul_buf[len++] = pkt->phy_data.header;

 memcpy(&ul_buf[len], pkt->phy_data.payload, pkt->phy_data_length);

 NBFi_Crypto_Encode(&ul_buf[len - 1], *((uint32_t*)Modem_ID), nbfi_iter.ul, 9);
 len += 8;
 mic_or_crc32 = NBFi_Crypto_UL_MIC(&ul_buf[len - 9], 9);
 nbfi_iter.ul = NBFI_Crypto_inc_iter(nbfi_iter.ul);
 ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 16);
 ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 8);
 ul_buf[len++] = (uint8_t)(mic_or_crc32);

 mic_or_crc32 = CRC32(ul_buf, 17);

 ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 16);
 ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 8);
 ul_buf[len++] = (uint8_t)(mic_or_crc32);

 if(nbfi.tx_freq)
 {
  tx_freq = nbfi.tx_freq ;
  parity = (nbfi.tx_freq > (nbfi.ul_freq_base));
 }
```

```
 else
 {
  tx_freq = NBFi_MAC_get_UL_freq(ul_buf[len - 4], parity);
 }

 for(int i=0; i<sizeof(protE_preambula); i++)
 {
  ul_buf_encoded[i] = protE_preambula[i];
 }

 PCODE_encode(8, &ul_buf[0], &ul_buf_encoded[4]);

 if(!nbfi.tx_freq) parity = !parity;


if((nbfi.additional_flags&NBFI_FLG_SEND_ALOHA) && parity)
{
 NBFi_RF_Init(nbfi.tx_phy_channel,(nbfi_rf_antenna_t)nbfi.tx_antenna,  nbfi.tx_pwr, tx_freq);

 NBFi_RF_Transmit(ul_buf_encoded, 36, nbfi.tx_phy_channel, BLOCKING);

 return NBFi_MAC_TX_ProtocolE(pkt);
}

 NBFi_RF_Init(nbfi.tx_phy_channel, (nbfi_rf_antenna_t)nbfi.tx_antenna,  nbfi.tx_pwr, tx_freq);

 NBFi_RF_Transmit(ul_buf_encoded, 36, nbfi.tx_phy_channel, NONBLOCKING);

 return OK;
}
```

E.2 Функция формирования и отправки DOWNLINK-пакета

```
nbfi_status_t NBFi_MAC_TX_ProtocolD(nbfi_transport_packet_t* pkt)
{
 uint8_t ul_buf[64];
 uint8_t len = 0;
 static _Bool parity = 0;
 uint8_t lastcrc8;
 uint32_t mic_or_crc32;

 nbfi_phy_channel_t phy;

 uint32_t tx_freq;
```

```
static uint32_t preamble;

memset(ul_buf,0,sizeof(ul_buf));

preamble = preambula(Modem_ID, (uint32_t *)0, (uint32_t *)0);
for(int i=0; i<sizeof(preamble); i++)
    ul_buf[len++] = ((uint8_t *)&preamble)[sizeof(preamble) - 1 - i];

ul_buf[len++] = nbfi_iter.ul;
ul_buf[len++] = pkt->phy_data.header;
memcpy(&ul_buf[len], pkt->phy_data.payload, pkt->phy_data_length);

NBFi_Crypto_Encode(&ul_buf[len - 1], Modem_ID, nbfi_iter.ul, 9);
len += 8;
mic_or_crc32 = NBFi_Crypto_UL_MIC(&ul_buf[len - 9], 9);
nbfi_iter.ul = NBFI_Crypto_inc_iter(nbfi_iter.ul);
ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 16);
ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 8);
ul_buf[len++] = (uint8_t)(mic_or_crc32);

mic_or_crc32 = CRC32(ul_buf + 4, 13);

ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 16);
ul_buf[len++] = (uint8_t)(mic_or_crc32 >> 8);
ul_buf[len++] = (uint8_t)(mic_or_crc32);

if(nbfi.tx_freq)
{
 tx_freq = nbfi.tx_freq ;
}
else
{
 tx_freq = NBFi_MAC_get_DL_freq();
}

ZCODE_Append(&ul_buf[4], &ul_buf[len], 1);


NBFi_RF_Init(nbfi.tx_phy_channel, (nbfi_rf_antenna_t)nbfi.tx_antenna, nbfi.tx_pwr, tx_freq);

NBFi_RF_Transmit(ul_buf, len + ZCODE_LEN, phy, NONBLOCKING);

return OK;
}
```

E.3 Функция вычисления контрольной суммы CRC8

```
static unsigned char CRC8byte(unsigned char data)
{
  uint8_t crc = 0;
  if(data & 1)     crc ^= 0x5e;
  if(data & 2)     crc ^= 0xbc;
  if(data & 4)     crc ^= 0x61;
  if(data & 8)     crc ^= 0xc2;
  if(data & 0x10)  crc ^= 0x9d;
  if(data & 0x20)  crc ^= 0x23;
  if(data & 0x40)  crc ^= 0x46;
  if(data & 0x80)  crc ^= 0x8c;
  return crc;
}

uint8_t CRC8(uint8_t* data, uint8_t len)
{
   uint8_t crc = 0;
   for(uint8_t i = 0; i < len; i++)
   {
      crc = CRC8byte(data[i] ^ crc);
   }
   return crc;
}
```

E.4 Функция вычисления контрольной суммы CRC16

```
#define POLY 0xa001
uint16_t CRC16(uint8_t *buf, uint16_t len, uint16_t crc)
{
   while (len--)
   {
      crc ^= *buf++;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
      crc = crc & 1 ? (crc >> 1) ^ POLY : crc >> 1;
   }
   return crc;
}
```

Е.5 Функция вычисления контрольной суммы CRC32

```
uint32_t CRC32(const uint8_t *buf, uint8_t len)
{
   return digital_update_crc32(0xffffffff, buf, len) ^ 0xffffffff;
}

#define WIDTH (8*4)
#define TOPBIT (1 << (WIDTH-1))
#define POLYNOMIAL (0x104C11DB7)

uint32_t crc_table(uint8_t n)
{
   uint32_t c;
   int k;
   c=((uint32_t)n) << (WIDTH - 8);
   for(k=8;k>0;k--)
   {
         if(c & (uint32_t)TOPBIT)
      {
         c = (c<<1) ^ POLYNOMIAL;
      }
      else
      {
         c=c<<1;
      }
   }
   return c;
}

uint32_t digital_update_crc32( uint32_t crc, const uint8_t *data, uint8_t len )
{
   while (len > 0)
   {
    crc = crc_table(*data ^ ((crc >> 24) & 0xff)) ^ (crc << 8);
    data++;
    len--;
   }
   return crc;
}
```

Приложение Ж
(обязательное)
Таблица и функции, используемые для ZIGZAG-кодирования данных

```
const uint8_t n[4][128] =
{

{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,3
4,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,6
5,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,9
6,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,11
9,120,121,122,123,124,125,126,127},

{104,52,43,96,31,7,71,78,58,37,93,25,125,85,42,111,6,95,72,117,27,51,63,84,91,35,120,26,97,4
5,110,70,1,28,86,114,53,67,12,127,40,101,73,94,115,61,20,126,3,46,92,116,9,56,87,77,109,44,6
5,54,100,118,2,34,21,41,76,14,69,124,90,18,103,48,113,36,0,81,13,62,24,38,105,68,15,75,88,50
,122,29,83,102,8,16,108,23,32,49,99,112,19,55,89,11,107,82,47,98,22,30,60,80,66,121,10,57,17
,39,79,4,64,123,33,59,106,74,5,119},

{26,10,105,48,38,84,76,57,23,125,115,3,106,33,77,99,71,113,22,1,44,87,8,31,111,96,2,42,70,81
,13,93,122,37,114,88,63,107,50,40,82,116,68,6,127,16,51,73,61,83,46,0,126,104,78,67,41,119,2
8,11,56,47,4,21,52,66,15,98,24,7,30,91,112,35,55,124,64,5,95,32,49,9,85,65,43,18,92,36,12,86,
118,60,25,72,53,80,123,45,58,102,110,120,89,34,17,75,94,27,100,62,20,39,108,90,69,117,97,59
,79,109,101,19,121,54,29,14,74,103},

{0,93,104,36,87,125,23,97,44,107,11,3,70,35,60,77,29,84,6,91,126,15,76,56,4,89,115,99,43,22,
122,16,105,55,2,113,78,51,63,14,120,102,8,19,68,111,86,47,64,32,121,72,59,108,96,80,25,67,1
18,12,58,127,20,90,9,37,103,53,62,69,85,10,110,34,100,119,39,73,1,83,48,112,30,54,65,45,5,12
3,101,26,88,18,46,95,40,109,7,27,57,66,116,38,75,92,21,52,61,28,106,114,94,33,17,79,42,71,12
4,50,82,13,31,41,117,74,98,81,24,49}
};


#define ZCODE_LEN 16

void ZCODE_Append(uint8_t * src_buf, uint8_t * dst_buf, _Bool parity)
{
 uint8_t b0;
 uint8_t b1;
 uint8_t bprev;
 uint8_t res;
 uint8_t code[4][8]=          {
                              {0,0,0,0,0,0,0,0},
                              {0,0,0,0,0,0,0,0},
                              {0,0,0,0,0,0,0,0},
```

```
                                {0,0,0,0,0,0,0,0}
                                };

for(int j=0; j<4; j++)
{
 for(uint8_t i=0; i<64; i++)
 {
  b0 = (src_buf[ n[j][ i ]  /8] << ( n[j][ i ]  %8)) & 0x80;
  b1 = (src_buf[ n[j][64+i] /8] << ( n[j][64+i] %8)) & 0x80;
  bprev   = (code[j][   (i-1)    /8] << (   (i-1)    %8)) & 0x80;
  res     =  b0 ^ b1 ^ bprev;
  code[j][i/8] |= res >> (i%8);
 }
}

for(int i=0; i<8;i++)
{
 dst_buf[i] = (code[0][i] & (parity ? 0xAA : 0x55)) | (code[1][i] & (parity ? 0x55 : 0xAA));
 dst_buf[i+8]    = (code[2][i] & (parity ? 0xAA : 0x55)) | (code[3][i] & (parity ? 0x55 : 0xAA));
}
}
```

Приложение И
(обязательное)
Исходные коды программной реализации помехоустойчивого кодирования

И.1 Исходные коды кодирования для сверточного кода

```
#define constraint_length 8
_Bool G_polys[2][8] = {{1,0,1,0,1,1,0,1}, {1,1,1,1,0,0,1,1}};
_Bool state[constraint_length]  = {0,0,0,0,0,0,0,0};

void conv_encoder(uint8_t* input, uint8_t* output, uint8_t len)
{
   _Bool* input_b  = (_Bool*)malloc(len * 8  * sizeof(_Bool));
   _Bool* output_d = (_Bool*)malloc(len * 2  * 8 * sizeof(_Bool));
   _Bool* output_b = (_Bool*)malloc(len * 2  * 8 * sizeof(_Bool));

   for (int n = 0; n < len * 8; n++)
   {
         uint8_t b = 7 - n % 8;
         input_b[n] = (input[n/8] >> b) & 1;
   }
   for (int n = 0; n < len * 8; n++)
   {
      _Bool input_d = input_b[n];
      _Bool u1 = input_b[n];
      _Bool u2 = input_b[n];

      for (int m = 1; m < constraint_length; m++)
      {
         if (G_polys[0][m])
            u1 = u1 ^ state[m-1];
         if (G_polys[1][m])
            u2 = u2 ^ state[m-1];
      }
      output_d[n * 2]     = u1;
      output_d[n * 2 + 1] = u2;

      for (int i = constraint_length; i > 0; i--)
         state[i] = state[i-1];

      state[0] = input_b[n];
   }
   uint16_t ptr_out = 0;
   for (int i = 0; i < 2 * len * 8; i++)
   {
```

```
            if (!(i % 10  == 3 || i % 10 == 8))
            {
                output_b[ptr_out] = output_d[i];
                ptr_out += 1;
            }
    }

    for (int i = 0; i < len; i++)
    {
        output[i] = 0;
        for (int b = 0; b < 8; b++)
        {
            output[i] |= output_b[i * 8 + b] << (7 - b);
        }
    }
    free(input_b);
    free(output_d);
    free(output_b);
}
```

И.2 Исходные коды кодирования для полярного кода

```
const uint8_t PCODE_indexes[160] = { 31, 47, 55, 57, 58, 59, 60, 61, 62, 63, 78, 79, 83, 85, 86,
87,89, 90, 91, 92, 93, 94, 95, 99, 101, 102, 103, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 135, 139, 141, 142, 143, 147,
149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 162, 163, 164, 165, 166, 167, 168, 169,
170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188,
189, 190, 191, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227,
228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
247, 248, 249, 250, 251, 252, 253, 254, 255 };


void PCODE_encode(uint8_t power, uint8_t* in, uint8_t* out)
{
 uint8_t data_20_i = 0;
 uint8_t data_20_sum[20];
 uint8_t t;
 uint32_t  sl;

 for (int i=0; i<20; i++)
 {
  data_20_sum[i] = 0;
  for (int j=0;j<8;j++)
  {
```

```
  data_20_sum[i] += (((in[i]>>(7-j) )& 1) * (1<<(7 - j)));
  data_20_i+= 1;
 }
}

for (int i=0; i<32; i++) out[i] = 0;

for (int i=0; i<160; i++)
{
 t = (data_20_sum[i/8] >> (7-(i%8))) & 1;
 out[PCODE_indexes[i]/8] |= t << (7 - PCODE_indexes[i] % 8);
}
for (int i=0; i < 32; i++)
{
 out[i] = out[i] ^ ((out[i] & 0x55) << 1);
 out[i] = out[i] ^ ((out[i] & 0x33) << 2);
 out[i] = out[i] ^ ((out[i] & 0x0F) << 4);
}
for (uint8_t i = 3; i < power; i++)
{
 sl = (1 << (i - 2));
 uint32_t temp = (1 << (power - 3)) / sl;
 for (uint32_t j=0; j < temp; j++)
 {
  for (uint32_t t = j*sl; t < j*sl+sl/2; t++)
  {
   out[t] = out[t] ^ out[t + sl/2];
  }
 }
}
}
```

Приложение К
(справочное)
Описание системы, использующей протокол NB-Fi

В настоящем приложении приведено описание информационной системы, использующей протокол NB-Fi на примере создания решения по дистанционному сбору данных с приборов учета энергоресурсов [4].

К.1 Архитектура системы

Архитектура системы состоит из четырех уровней и включает в себя:

1) оконечные устройства (приборы учета энергоресурсов), отправляющие и принимающие данные по протоколу стандарта NB-Fi;

2) базовые станции, осуществляющие прием, обработку и передачу сообщений от устройств (UPLINK-пакеты) на сервер NB-Fi, а также отправку нисходящих сообщений на устройства (DOWNLINK-пакеты);

3) NB-Fi сервер, осуществляющий прием, обработку и хранение сообщений от всех базовых станций для всех устройств, а также обеспечивающий интеграцию данных с сервером приложений и со сторонними программно-техническими комплексами (информационно-вычислительными комплексами верхнего уровня);

4) сервер приложений, осуществляющий отображение полученных данных от устройств для заказчиков, и предоставляющий возможность выгрузки отчетов в требуемом виде.

К.2 Описание устройств

Устройства, используемые для сбора данных с приборов учета ЖКХ, построены с использованием интегрального радиотрансивера WA1470, поддерживающего аппаратную реализацию физического и MAC-уровня протокола NB-Fi. Транспортный уровень протокола NB-Fi реализован при помощи программной библиотеки, исполняемой на микроконтроллере STM32L0x. Библиотека NB-Fi и примеры ее использования, содержатся в файловом архиве [5].

Система содержит оконечные устройства NB-Fi, обеспечивающие учет потребления коммунальных ресурсов и сбор данных с существующих приборов учета, среди которых:

- счетчики воды с накладным внешним модемом;
- счетчики воды со встроенным в основную плату модемом;
- однофазные и трехфазные счетчики электрической энергии со встроенным модемом;
- теплосчетчики, со встроенным и внешним модемом;
- счетчики газа, со встроенным модемом;
- радиомодемы, подсоединяющиеся к импульсным или цифровым выводам внешних устройств.

Перечисленные выше устройства имеют стационарное питание (счетчики электроэнергии) либо питаются от встроенной батареи (счетчики воды, газа, тепла, внешние модемы). При этом, ввиду невысоких требований к объемам передаваемых данных, емкости встроенной батареи достаточно для электропитания устройств на протяжении всего срока жизни.

Счетчики воды, газа, тепла отправляют сообщения два раза в сутки и содержат внутри себя данные о почасовом потреблении энергоресурсов.

Счетчики воды с накладным модемом получают показания о потреблении при помощи оптического датчика, который считывает вращение бегунка. Накопленное число вращений передается на сервер приложений, где складывается с начальными показаниями данных счетчика.

Аналогично работает радиомодем, считывающий импульсы с импульсных выходов устройств: накопленные значения импульсов передаются на сервер приложений, где суммируются с начальными показаниями.

Счетчики воды, тепла, газа со встроенным радиомодулем два раза в сутки передают актуальные показания с почасовой разбивкой. Счетчики со встроенным модемом имеют жидкокристаллический дисплей, отображающий показания, и именно эти показания и передаются на сервер приложений.

Счетчики тепла, электроэнергии с внешним радиомодемом с цифровым (RS-485) выходом считывают показания с прибора учета по цифровому интерфейсу. Для отдельных приборов учета разработаны индивидуальные радиомодемы, позволяющие осуществить монтаж радиомодема под крышку устройства. Радиомодемы с батарейным питанием, в зависимости от конфигурации, отправляют данные с периодичностью от одного раза в месяц до двух раз в сутки. Существует модель радиомодема для электросчетчика с питанием от сети, что позволяет увеличить срок службы радиомодема.

Указанные выше устройства работают в режимах NRX либо DRX.

Счетчики электроэнергии осуществляют отправку сообщений на сервер один раз в час или чаще, в зависимости от конфигурации. Сообщения содержат данные о потреблении электроэнергии по каждому тарифу, а также другие параметры сети, в зависимости от настроек. Устройства работают в режиме CRX (Continuous RX) и обеспечивают постоянный прием нисходящих (DOWNLINK) сообщений от сервера. Отправка данных на электросчетчик с сервера возможна в любой момент. Данный режим работы конфигурировать электросчетчик и осуществлять управление размыкающего реле.

При работе в режиме NRX (а также в режимах DRX и CRX при низком уровне радиосигнала) возможны потери отправленных пакетов. В случае пропуска данных в определенный период, на сервере приложений данных за этот период не окажется. Для режимов DRX и CRX пропущенные данные могут быть впоследствии запрошены повторно сервером приложений.

Мощность излучения устройств, составляет 25 мВт при максимально разрешенном уровне излучения 100 мВт (для нелицензируемого диапазона частот 868,7 – 869,2 МГц). Расчетный срок работы устройств от батареи составляет свыше 10 лет. Рабочий температурный диапазон устройств составляет от минус 40 °С до плюс 70 °С.

К.3 Описание базовых станций

Базовая станция NB-Fi является оборудованием базовых станций сетей радиодоступа и предназначено для приема-передачи маломощного радиосигнала узкополосной беспроводной технологии связи в субгигагерцовом диапазоне радиочастот. Базовая станция NB-Fi обеспечивает прием и передачу информации посредством радиоэфира с приборами учета энергоресурсов, с радиомодемами, подсоединенными к приборам учета энергоресурсов, с прочими датчиками (далее – устройствами), работающими в пределах

рабочей частоты приемника и передатчика, и передачу этой информации на сервера и информационно-вычислительные комплексы верхнего уровня автоматизированных систем (далее – ИВК ВУ) через стандартные интерфейсы и каналы связи, в том числе, по сети Интернет или посредством изолированных локальных сетей.

Для приема восходящих пакетов данных (UPLINK-пакетов) со стороны Базовой станции NB-Fi применяется принцип SDR-систем (Software-Defined Radio), где входной радиосигнал оцифровывается во всей полосе приема 51,2 кГц, и в дальнейшем подвергается программной обработке. Прием пакетов выполняется Базовой станцией NB-Fi по всей полосе частот, при этом выделяются одновременно пакеты, имеющие различные скорости. Теоретическое количество каналов составляет 1024 для скорости 50 бит/с, и 128, 16 и 1 для скоростей 400, 3200 и 25600 бит/с соответственно.

Передача нисходящих пакетов данных (DOWNLINK-пакетов) выполняется при помощи цифрового модулятора сигнала, позволяющего передавать одновременно несколько узкополосных каналов, если суммарная мощность передачи не превышает установленного значения. Передача осуществляется в полосе 102,4 кГц.

Для работы сети передачи данных по протоколу NB-Fi в Российской Федерации используется нелицензируемый в Российской Федерации диапазон частот 868,7 – 869,2 МГц. Максимальная выходная мощность передачи Базовой станции NB-Fi составляет 100 мВт.

В некоторых вариантах реализации Базовая станция NB-Fi может обладать следующими характеристиками:

- Рабочий температурный диапазон составляет от минус 50 °С до плюс 70 °С.

- Степень защиты корпуса от проникновения пыли и воды соответствует IP66 по ГОСТ 14254 (IEC 60529:2013).

- Сервер NB-Fi и сервер приложения могут быть реализованы прямо на Базовой станции NB-Fi, что обуславливается требованиями заказчиков к автономности и изолированности используемых систем.

- За счет разнесения полос приема и передачи по частоте и развязки приемной и передающих антенн по поляризации, Базовые станции NB-Fi обеспечивают одновременный прием и передачу данных (режим передачи данных «полный дуплекс») без ухудшения характеристик радиосвязи.

- Базовые станции NB-Fi обеспечивают прием и передачу данных по одному каналу связи, но с разделением по времени (режим передачи данных «полудуплекс»), могут иметь более низкий динамический диапазон приемной части, и меньшую чувствительность приема.

К.4 Описание сервера NB-Fi

Функциями сервера NB-Fi являются:

- хранение в базе данных информации об устройствах (идентификаторы, ключи шифрования MAC-уровня, режимы работы транспортного уровня);
- получение восходящих пакетов с данными от множества базовых станций;
- дешифрование данных MAC-уровня и отбрасывание пакетов, не прошедших проверку целостности данных;
- выделение уникальных пакетов (фильтрация и отбрасывание копий одного и того же пакета, принятого разными станциями);
- реализация функций транспортного уровня NB-Fi;
- выбор наилучшей базовой станции для отправки нисходящих пакетов для каждого устройства;
- сохранение пакетов в базу данных;
- взаимодействие с серверным программным обеспечением уровня приложений посредством API-интерфейса.

Использование центрального сервера NB-Fi в качестве ответного узла для всех оконечных устройств сети NB-Fi позволяет организовать сеть передачи данных большого радиуса действия, содержащей большое количество устройств (от сотен тысяч до десятков миллионов). Преимуществом реализации транспортного уровня на стороне сервера является простое разрешение коллизий между пересекающимися по покрытию базовыми станциями, при отправке нисходящих пакетов на устройства. В подобной централизованной архитектуре масштабирование сети по расширению покрытия либо по увеличению плотности установки устройств выполняется максимально легко, простым добавлением базовых станций и устройств, без необходимости конфигурирования.

Установка сервера NB-Fi возможна в том числе и на Базовые станции NB-Fi. Это позволяет организовать получение/отправку данных непосредственно с базовых станций и на них. При этом возникает необходимость конфигурирования каждой станции в части загрузки в нее информации об устройствах, что усложняет масштабирование сети.

К.5 Описание сервера приложений

Функциями сервера приложений являются:

- поддержка моделей оконечных устройств:

а) преобразование пакетов от сервера NB-Fi в данные, пригодные для отображения пользователю устройств;

б) преобразование команд от пользователя устройств в пакеты для отправки через сервер NB-Fi;

в) конфигурирование устройств, обновление внутреннего программного обеспечения устройств, диагностика исправностей;

- длительное хранение данных;
- предоставление данных и управление устройствами через API-интерфейс;
- предоставление данных и управление устройствами при помощи графического интерфейса пользователя;
- аналитический и статистический анализ данных, выгрузка отчетов;

- другие функции, зависящие от предметной области данного приложения.

Сервер приложений является верхним уровнем телекоммуникационной системы, построенной на базе сети передачи данных NB-Fi. Данный компонент не является стандартным решением со строго определенным перечнем функций. Его задачи зависят от предметной области, для которой используется система (автоматизированная система коммерческого учета энергоресурсов, системы мониторинга и сигнализации, контроль технологических процессов для промышленности и сельского хозяйства и т.д.). Возможна одновременная работа различных серверов приложения, взаимодействующих с единым сервером NB-Fi.

8 Библиография

[1] Золотарёв В.В., Овечкин Г.В. Помехоустойчивое кодирование. Методы и алгоритмы: Справочник – М.: Горячая линия–Телеком, 2004.

[2] Erdal Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels” (Метод построения кодов, достигающих максимальноsй емкости, для симметричных двоичных каналов), URL: https://arxiv.org/abs/0807.3917, дата обращения: 25.09.2020

[3] Li Ping, Xiaoling Huang, and Nam Phamdo, “Zigzag Codes and Concatenated Zigzag Codes” (Зигзагообразные коды и составные зигзагообразные коды), URL: https://pdfs.semanticscholar.org/3091/6eb78443174658fbf8d4464a5bf966846399.pdf, дата обращения: 25.09.2020

[4] Проекты компании ООО «Телематические Решения», URL: https://waviot.ru/solutions/ , дата обращения: 25.09.2020

[5] Корпоративный файловый репозиторий компании ООО «Телематические Решения», URL: https://github.com/waviot/NBFi_WA1470, дата обращения: 25.09.2020

УДК 004.738 ОКС 35.020, 35.110

Ключевые слова: информационные технологии, интернет вещей, протокол беспроводной передачи данных, узкополосная модуляция радиосигнала, NB-Fi