Архитектура компьютерных сетей

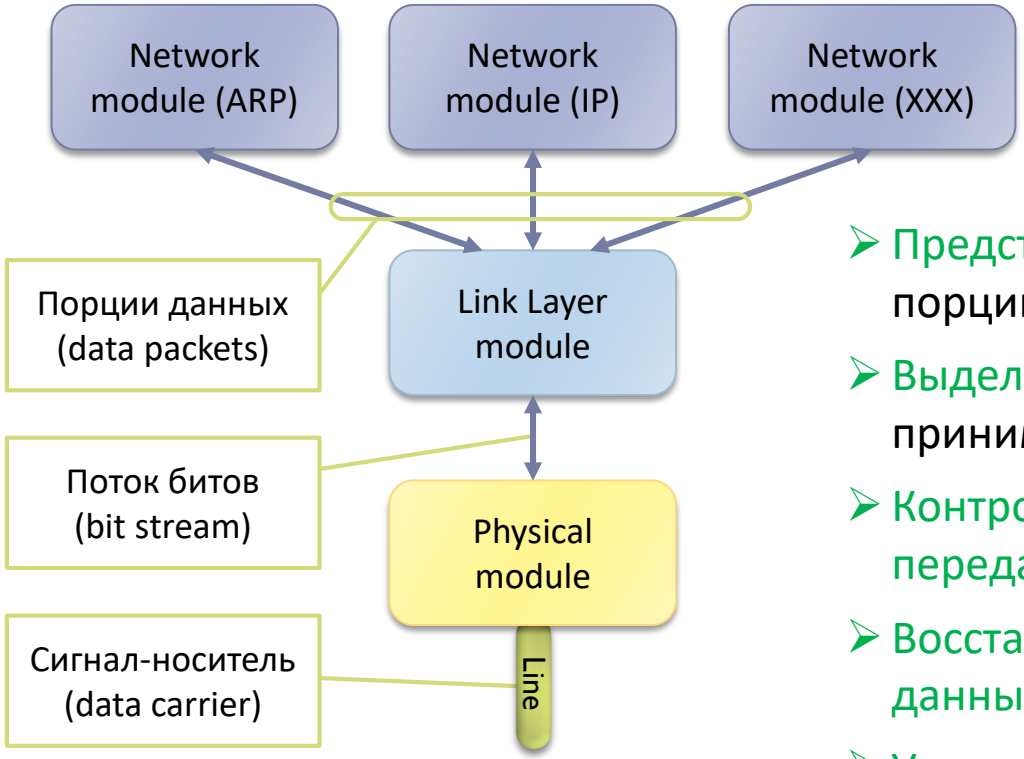
Останков Александр Иванович

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. **Архитектура модулей канального уровня**
    - 6.1. Функция кадрирования
    - 6.2. Методы контроля правильности передачи данных
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - 6.4. Управление звеном передачи данных
    - 6.5. Семейство протоколов HDLC
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

Назначение и функции канального уровня

Роль канального уровня в протокольном стеке – доставлять порции данных (пакеты) между **соседними** узлами, подключенными к общему звену передачи



Для этого канальному модулю необходимо уметь:

- Представлять последовательность порций данных в виде потока битов
- Выделять порции данных из принимаемого потока битов
- Контролировать правильность передачи порций по звену
- Восстанавливать ошибочно принятые данные
- Управлять очередностью доступа к разделяемой среде передачи данных

Классификация используемых звеньев

По конфигурации звена передачи данных:

- точка-точка (point to point, PPP)
- многоточечные (multipoint)

По режиму передачи:

- симплексные – передача возможна только в одном направлении
- дуплексные – прием возможен одновременно с передачей
- полудуплексные – направление передача/прием может переключаться

По способу организации потока передаваемых данных:

- побитовые синхронные - последовательность битов передаваемых друг за другом без пауз с постоянной скоростью
- посимвольные синхронные - последовательность символов фиксированного размера (октет - 8 бит) передаваемых друг за другом без пауз с постоянной скоростью
- посимвольные асинхронные – последовательность символов фиксированного размера (октет – 5..9 бит) между передачей которых допускаются паузы произвольной длины

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - **6.1. Функция кадрирования**
    - 6.2. Методы контроля правильности передачи данных
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - 6.4. Управление звеном передачи данных
    - 6.5. Семейство протоколов HDLC
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

Проблема кадрирования (Framing)

- ✓ У приемника нет сведений о том, что в данный момент передается по линии и какого размера должны быть пакеты. Все чем располагает приемник – это принимаемая последовательность данных
- ✓ Пакет может содержать любую произвольную комбинацию битов, поэтому если передавать их подряд друг за другом, то на приеме невозможно будет определить где заканчивается один и начинается другой пакет
- ✓ Поэтому передатчик должен сформировать поток данных так, чтобы приемник смог разделить его на исходные порции – кадры (frames), анализируя только данные принимаемого потока битов

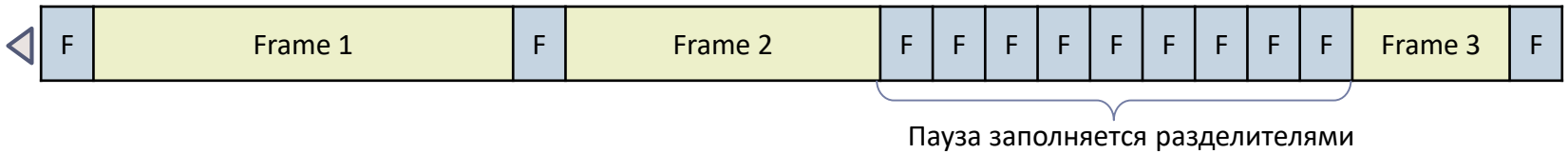
Как это можно сделать ?

Использование разделителя кадров

Разделитель это определенная комбинация в потоке данных. Ею могут быть:

- пауза в потоке (там где это возможно на физическом уровне)
- символ-разделитель (Flag = 01111110, 0x7E)

Структура потока с разделителями:



Кадр – это последовательность битов между двумя разделителями

Разделитель не должен появляться внутри кадра !!!! Для достижения кодопрозрачности передатчик модифицирует передаваемые данные:

- вставкой битов (bit-stuffing)
- вставкой символа-модификатора (control-escape)

Кодопрозрачность путем вставки битов

Правило работы передатчика:

Вставлять дополнительный **0** после **пяти** единиц подряд

Правило работы приемника:

Удалять **0** из любой последовательности **…111110…**

Комбинация флага

Данные: 0100110101**01111110**1111100111011111111111110000..

Передаются: 0100110101011111**0**10111110**0**0111011111**0**11111**0**1110000..

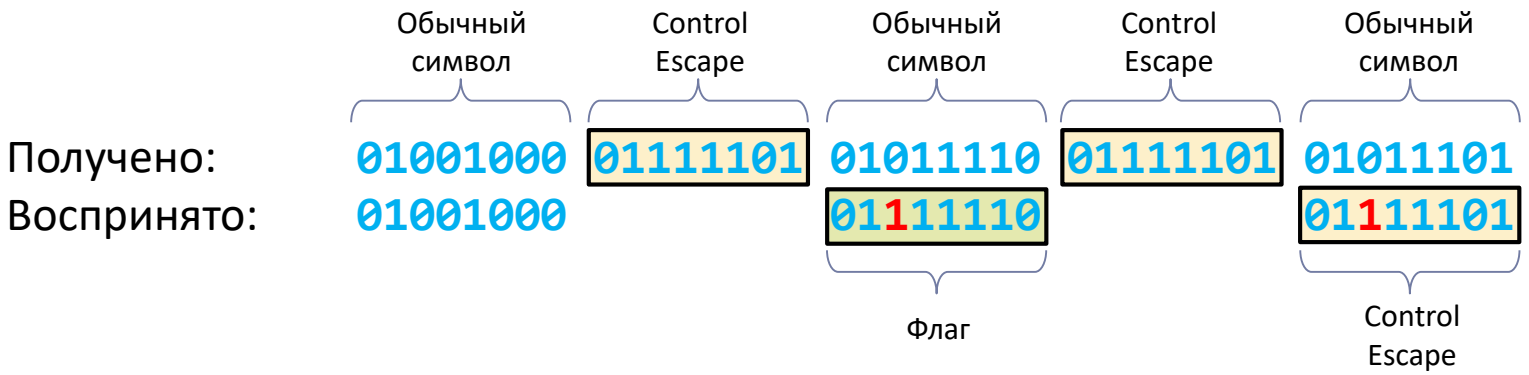
- ✓ Поскольку в комбинации (01111110) **шесть** единиц подряд случайное появление флага в модифицированном потоке битов исключено
- ✓ Bit-stuffing легко и эффективно реализуется **аппаратно** на синхронных **побитовых** каналах. Однако его **программная** реализация оказывается слишком медленной.
- ✓ Bit-stuffing невозможно применять на **посимвольных** каналах.

Использование символа-модификатора

Посимвольный физический канал передает последовательность символов (октетов) фиксированной длины (обычно 8 бит).

Поэтому для реализации кодопрозрачности приходится вставлять целый символ (Control-Escape = **01111101, 0x7D**), а не один бит.

Появление этого символа в потоке предписывает приемнику модифицировать следующий символ путем инвертирования третьего бита (**00100000**):



Передатчик должен модифицировать все «особые» символы кадра (как минимум символы-флаги и символы-модификаторы) со вставкой символа-модификатора перед ними. Так «особые» символы в потоке преобразуются в «обычные» символы.

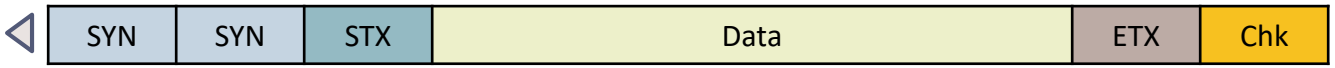
Другие способы кадрирования

С указателем длины кадра:

практически не применялась - при искажении величины указателя длины возможна длительная потеря кадровой синхронизации.

С применением управляющих символов (байт-ориентированное кадрирование):

исторический способ (1960-70) при котором структура кадров выделялась специальными управляющими символами ASCII, например:



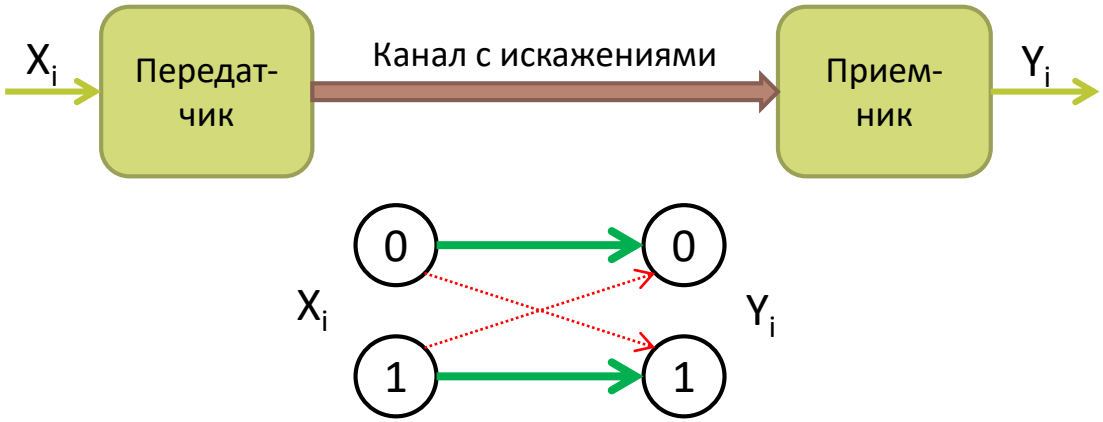
Код	Шифр	Назначение
02	STX	Start of text – начало данных кадра
03	ETX	End of text – конец данных кадра
10	DLE	Data Link Escape – символ «модификатор»
32	SYN	Synchronous idle – символ-заполнитель паузы
….	…..	……..

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - 6.1. Функция кадрирования
    - **6.2. Методы контроля правильности передачи данных**
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - 6.4. Управление звеном передачи данных
    - 6.5. Семейство протоколов HDLC
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

Контроль правильности передачи

Любые реальные каналы связи вносят ошибки в передаваемые данные → обнаружение (и исправление) ошибок – одна из главных задач Netware.



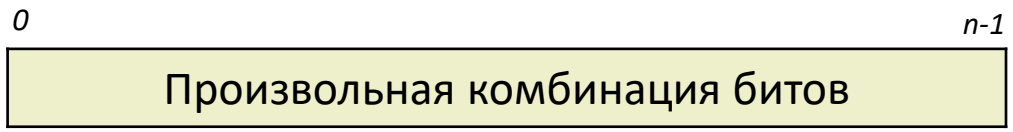
Если $X_i = Y_i$ ошибки нет. При $X_i \neq Y_i$ ошибка присутствует.

Интенсивность ошибок (bit error rate) $BER = \frac{Колич.ош.бит}{Всего\ бит}$

При BER=0.5 – величина Y_i вообще не связана с X_i - такой канал не способен передавать информацию

Принцип обнаружения ошибок

Кодовое слово ***(D)*** – порция передачи информации по каналу (длиной n бит):



Кодовая комбинация (d^j) - уникальное сочетание значений битов кодового слова. Всего возможно 2^n различных кодовых комбинаций.

Кодовое пространство ***(Δ)*** – совокупность всех возможных кодовых комбинаций.

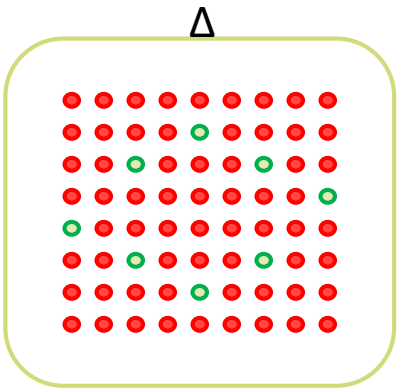
В случае ошибки в канале передаваемая кодовая комбинация (d^s) будет принята как другая кодовая комбинация (d^e) при этом обе эти комбинации будут принадлежать ***(Δ)***.

Функция обнаружения ошибки предполагает, что приемник способен зафиксировать, что полученная кодовая комбинация (d^e) не соответствует переданной (d^s), **не зная, при этом, что именно было передано** !

Метод обнаружения ошибок

Все кодовое пространство (Δ) делится на два подмножества:

- допустимые кодовые комбинации $\{d^+\}$: «»
- недопустимые кодовые комбинации $\{d^-\}$: «»



Передатчик имеет право использовать (передавать) только комбинации из $\{d^+\}$. Таким образом, если приемник получит на приеме комбинацию из $\{d^-\}$ - то это ошибка !

Однако, если в результате искажения одна кодовая комбинация из $\{d^+\}$ превратится в другую допустимую комбинацию из $\{d^+\}$, то такая ошибка останется не обнаруженной !!!!

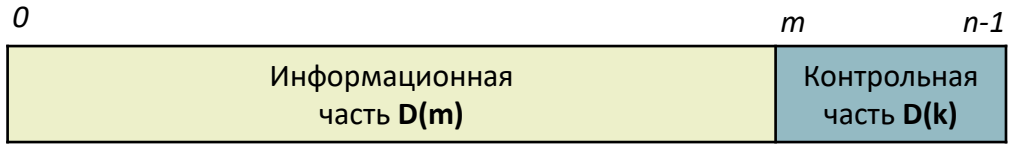
Вероятность не обнаружения ошибки, прежде всего, зависит от соотношения количества комбинаций в $\{d^+\}$ к количеству комбинаций в $\{d^-\}$. Пример:

- количество допустимых комбинаций 8
- количество недопустимых комбинаций 56

Пусть вероятность перехода в любое другое состояние одинакова: 1/63, тогда вероятность перехода в $\{d^+\}$ = 7/63, а в $\{d^-\}$ = 56/63. Итого ошибка не обнаружится только в 7/56 = 1/8 случаев

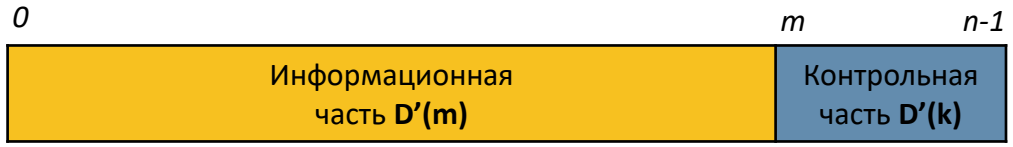
Способ обнаружения ошибок

Кодовое слово **(*D*)** делится на две части: ***D(m)*** (**m** бит) и ***D(k)*** (**k = n-m** бит):



- ***D(m)*** определяется передаваемыми данными
- ***D(k)*** определяется передатчиком по известной функции: $D(k) = f^{chk}(D(m))$

В итоге передатчик передаст по каналу одну из 2^m допустимых комбинаций из 2^n возможных ($n = m+k$), а приемник получит кодовое слово **(*D'*)**, возможно искаженное в канале



Приемник по принятым данным ***D'(m)*** рассчитывает $D''(k) = f^{chk}(D'(m))$

Если принятая и рассчитанная контрольные части не совпадают ($D'(k) \neq D''(k)$), то **фиксируется факт ошибки**

Эффективность схем обнаружения ошибок

Никакая схема не способна обнаруживать все 100% ошибок !!!!

Тем не менее, возможно достичь сколь угодно высокой вероятности обнаружения ошибок ценой введения большей избыточности: $(\gamma = k / m)$

Однако эффективность применяемых схем зависит не только от вносимой избыточности.

Метрики эффективности схем обнаружения:

Кодовое расстояние между двумя кодовыми комбинациями (d^s) и (d^e) – количество битов, которые необходимо инвертировать в (d^s), чтобы получить (d^e)

Обнаруживающая способность схемы – минимальное кодовое расстояние между двумя допустимыми кодовыми комбинациями

(Оценка минимального кодового расстояния полезна тем, что вероятность возникновения ошибок малой кратности существенно выше многократных)

Пример схемы обнаружения ошибок: полное дублирование передаваемых данных.

Контроль по паритету

Контроль по нечетности (odd parity)

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	c
0	1	0	0	1	0	1	0	**0**
0	1	1	1	0	0	1	1	**0**
1	1	1	0	0	1	0	0	**1**
0	0	1	1	0	0	0	0	**1**
0	0	1	1	1	1	0	1	**0**

Контроль по четности (even parity)

0	1	0	0	1	0	1	0	**1**
0	1	1	1	0	0	1	1	**1**
1	1	1	0	0	1	0	0	**0**
0	0	1	1	0	0	0	0	**0**
0	0	1	1	1	1	0	1	**1**

$$c = b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7$$

Свойства контроля по паритету

Исходное кодовое слово - (odd)	0	1	0	0	1	0	1	0	**0**
				X					
Полученное кодовое слово – (even)	0	1	0	1	1	0	1	0	**0**

Исходное кодовое слово - (odd)	1	1	0	0	1	1	0	0	**1**
				X		X			
Полученное кодовое слово – (**odd**)!	1	1	0	1	1	0	0	0	**1**

Избыточность: **(*n,n-1*)** – минимальна, всего 1 бит
Кодовое расстояние: **2**
Позволяет обнаруживать: **одинарные** и нечетнократные ошибки
Не обнаруживает: **двойные** и четнократные ошибки

Матричные коды

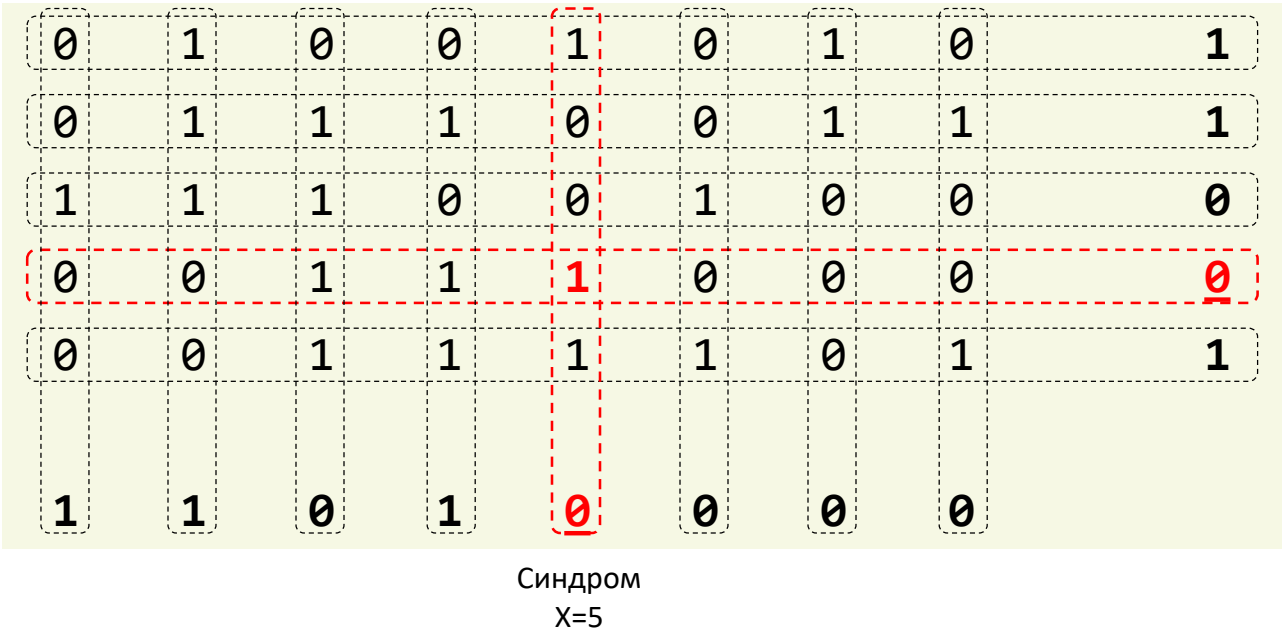
Ортогональный матричный код (even)

0	1	0	0	1	0	1	0	**1**
0	1	1	1	0	0	1	1	**1**
1	1	1	0	0	1	0	0	**0**
0	0	1	1	0	0	0	0	**0**
0	0	1	1	1	1	0	1	**1**
1	**1**	**0**	**1**	**0**	**0**	**0**	**0**	

Избыточность: ***(53,40)*** – 13 бит
Кодовое расстояние: **3**
Позволяет обнаруживать: **одинарные** и **двойные** ошибки
Не обнаруживает: часть ошибок **кратности 3** и выше

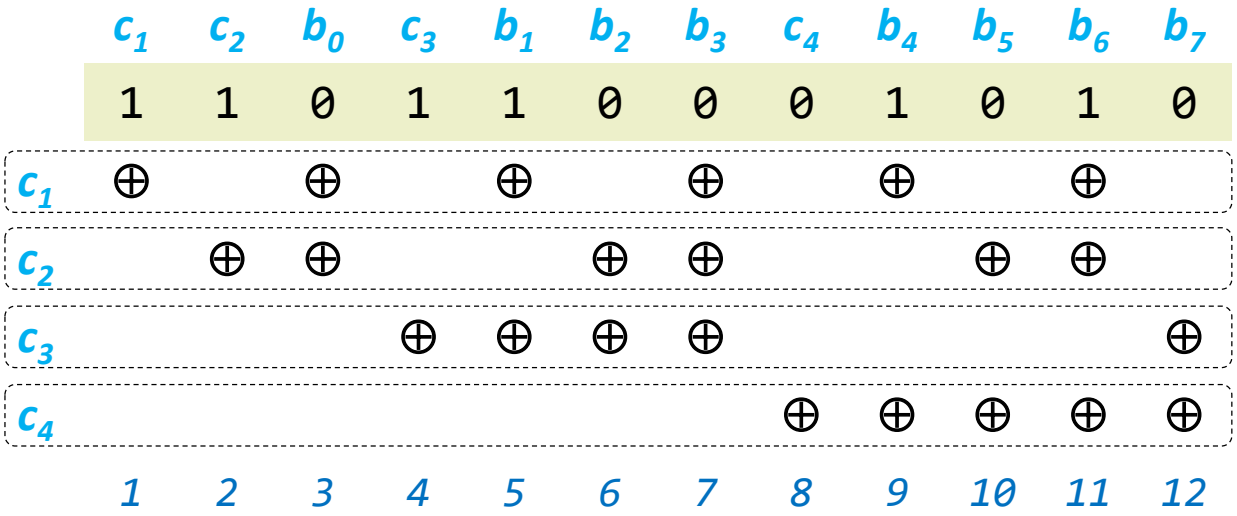
Исправляющая способность

Ортогональный матричный код (even)



При одиночной ошибке синдромы указывают на координату ошибочного бита в матрице. Эта информация позволяет исправить ошибку (инвертировать значение ошибочного бита)

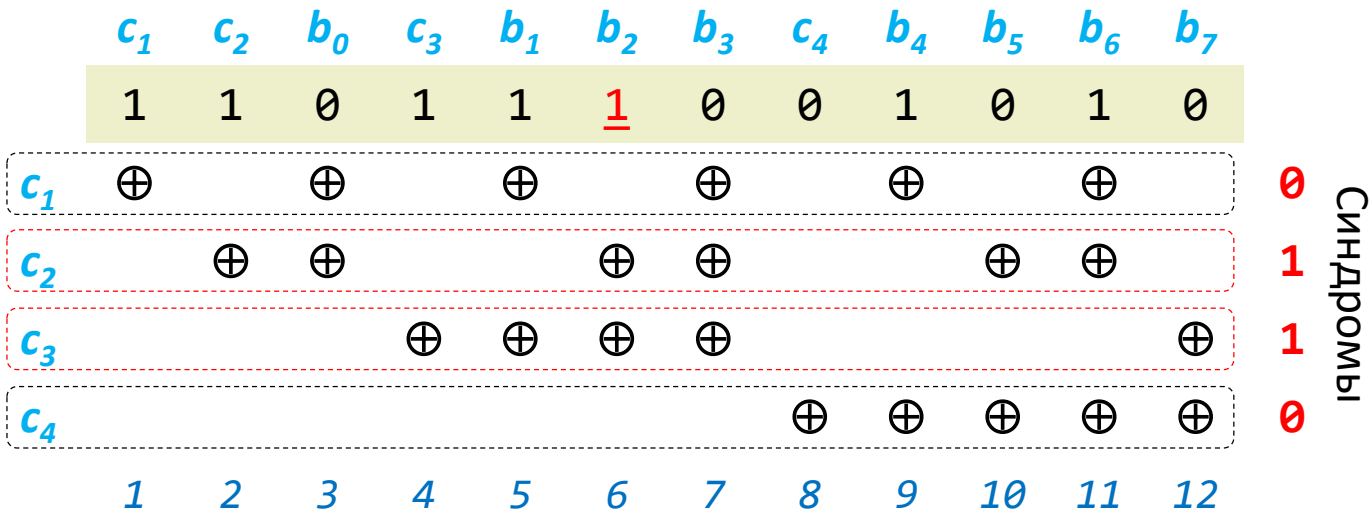
Код Хэмминга



Избыточность:	**(*12,8*)** – 4 бит
Кодовое расстояние:	**3**
Позволяет обнаруживать:	**одинарные** и **двойные** ошибки
Не обнаруживает:	часть ошибок **кратности 3** и выше

Полный код Хэмминга содержит 2^k-*1* бит и способен кодировать 2^k-*k*-*1* битов данных: (7,4), (15,11), (31,26), (63,57), (127,120) и т.п. Применяют и усеченные коды, например (71,64). Полный код является совершенным кодом.

Исправляющая способность кода Хэмминга



При одиночной ошибке комбинация значений синдромов в коде Хэмминга указывает на номер ошибочного бита ($0110_2 = 6$). Однако двойные ошибки также приводят к генерации синдромов, которые уже невозможно исправить.

Модифицированный код Хэмминга (с добавлением дополнительного бита общего паритета) увеличивает кодовое расстояние до 4 и позволяет различать одиночные от двойных ошибок. В этом варианте код способен исправлять одиночные и обнаруживать все двойные ошибки.

Модифицированный код (72,64) широко используется как основа для **ECC RAM**.

Кодовое расстояние и способность кода обнаруживать и исправлять ошибки

Теорема об обнаружении ошибок

Код с минимальным кодовым расстоянием d_{min} может быть способен к обнаружению до p ошибок в том и только в том случае, если

$$d_{min} \geq p+1$$

Теорема об исправлении ошибок

Код с минимальным кодовым расстоянием d_{min} может быть способен к исправлению до t ошибок и к обнаружению дополнительных p ошибок в том и только в том случае, если

$$d_{min} \geq 2t+p+1$$

Альтернативы паритетным кодам

Недостатки паритетных кодов:

- Имеют невысокую обнаруживающую способность (максимум 4)
- Не подходят для обнаружения всплесков ошибок (error bursts)

Основные причины:

- на величину каждого контрольного бита (паритетной формулы) влияет лишь подмножество битов, а не все биты кодового слова
- каждая паритетная формула имеет лишь два равновероятных состояния (паритет соответствует или не соответствует)
- каждая паритетная формула рассчитывается изолированно от других

Метод устранения недостатков – вычисление многобитового контрольного числа (контрольной суммы - chksum), величина которого должна зависеть от всех битов кодового слова.

Каким способом следует вычислять контрольное число ?

Варианты вычисления контрольных чисел

- Простое суммирование слов длиной ***k*** (арифметическое по модулю 2^k или побитовое – XOR):

$$D(k) = (D(m)_{[1\ldots k]} + D(m)_{[k+1\ldots 2k]} + \ldots + D(m)_{[m-k+1\ldots m]})_{mod\ 2^k}$$

  Обнаруживающая способность всего 1 (!!!!) – не подходит.

- Позиционно-зависимое суммирование:

$$D(k) = (\alpha_1 * D(m)_{[1\ldots k]} + \alpha_2 * D(m)_{[k+1\ldots 2k]} + \ldots + \alpha_l * D(m)_{[m-k+1\ldots m]})_{mod\ 2^k}$$

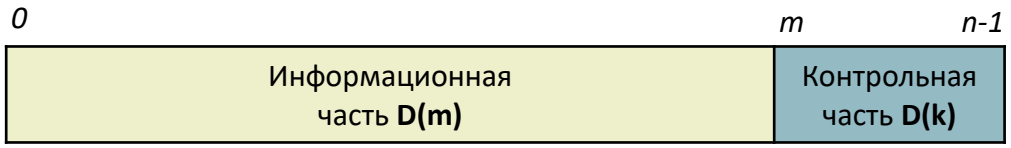
  Лучше, но многократное умножение реализовать непросто и разные биты ***D(m)*** оказывают разное влияние на величину ***D(k)***, особенно при больших величинах α_i

- Использование свойства делимости (кратности):

  $D = D(m)*2^k + D(k)$ без остатка делится на ***G*** (известную константу)

В 1961 году W. Wesley Peterson изобрел эффективный способ вычисления ***D(k)*** и проверки на делимость – циклический избыточный код, cyclic redundancy check (**CRC**).

Принцип Cyclic redundancy code (CRC)



Для каждой комбинации ***D(m)*** подбирается ***D(k)*** таким образом, чтобы все кодовое слово $D = D(m) \cdot 2^k + D(k)$ **делилось без остатка** на некоторую константу ***G***
Если $2^{k-1} < G < 2^k$, то для каждой комбинации ***D(m)*** будет только одна ***D(k)***

Деление по правилам обычной арифметики это слишком сложная операция, поэтому применяется правила **арифметики двоичных многочленов** вида:

$$d = \delta_{n-1} \cdot x^{n-1} + \delta_{n-2} \cdot x^{n-2} + \ldots + \delta_1 \cdot x^1 + \delta_0 \cdot x^0$$

Здесь δ_i это коэффициенты многочлена ***{0,1}*** соответствующие битам ***D*** и при вычислении используется **двоичная арифметика без переносов**:

0 + 0 = 0	0 – 0 = 0	0 · 0 = 0
0 + 1 = 1	0 – 1 = 1	0 · 1 = 0
1 + 0 = 1	1 – 0 = 1	1 · 0 = 0
1 + 1 = 0	1 – 1 = 0	1 · 1 = 1

Для двоичных многочленов определены арифметические операции: сложение, умножение, вычитание (эквивалентно сложению), а также **деление с остатком**

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$	$1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$
$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3$	$1 \cdot x^3$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3$ | $1 \cdot x^3$

$1 \cdot x^4 + 1 \cdot x^3$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$\underline{1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3}$ | $1 \cdot x^3$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$\underline{1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3}$ | $1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3$ | $1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^3 + 0 \cdot x^2$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$\underline{1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3}$ | $1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$\underline{1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2}$

$1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$\underline{1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3}$ | $1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$\underline{1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2}$

$1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1$

$1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3$ | $1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1$

$1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^1$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 \;\big|\; 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$\underline{1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3} \;\big|\; 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$\underline{1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2}$

$1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1$

$\underline{1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1}$

$1 \cdot x^1 + 1 \cdot x^0$

Деление многочленов «в столбик»

$1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$ | $1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0$

$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3$ | $1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2$

$1 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1$

$1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1$

$1 \cdot x^1 + 1 \cdot x^0$

Частное интереса не представляет, тогда как остаток

$$(1 \cdot x^1 + 1 \cdot x^0) = mod((1 \cdot x^5 + 1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0), (1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0))$$

используется при построении кодового слова с CRC по правилу:

$$D = D(m) \cdot x^k + D(k),$$ где $$D(k) = mod(D(m) \cdot x^k, G)$$

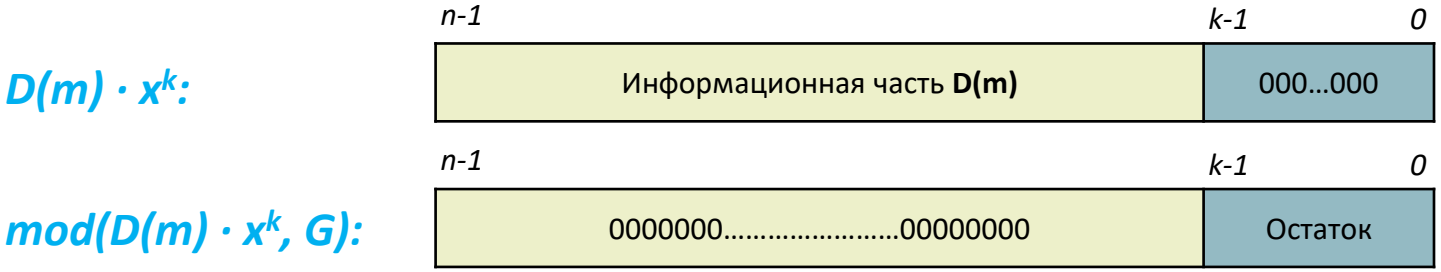
здесь G – многочлен-константа степени k, называемый **образующим полином**

Формирование CRC

Образующий полином должен быть степени ***k*** (k – количество контрольных битов в кодовом слове):

$$G = \underline{1} \cdot x^k + \gamma_{k-1} \cdot x^{k-1} + \ldots + \gamma_0 \cdot x^0$$

Благодаря этому остаток будет содержать максимум ***k*** битов.



Сформированный код будет делиться без остатка на G:

$D(m) \cdot x^k = Q \cdot G + mod(D(m) \cdot x^k, G)$, где Q - это частное от $D(m) \cdot x^k \ / \ G$

$D(m) \cdot x^k + mod(D(m) \cdot x^k, G) = Q \cdot G + mod(D(m) \cdot x^k, G) + mod(D(m) \cdot x^k, G)$

Но, по правилам арифметики двоичных многочленов

$mod(D(m) \cdot x^k, G) + mod(D(m) \cdot x^k, G) = 0$

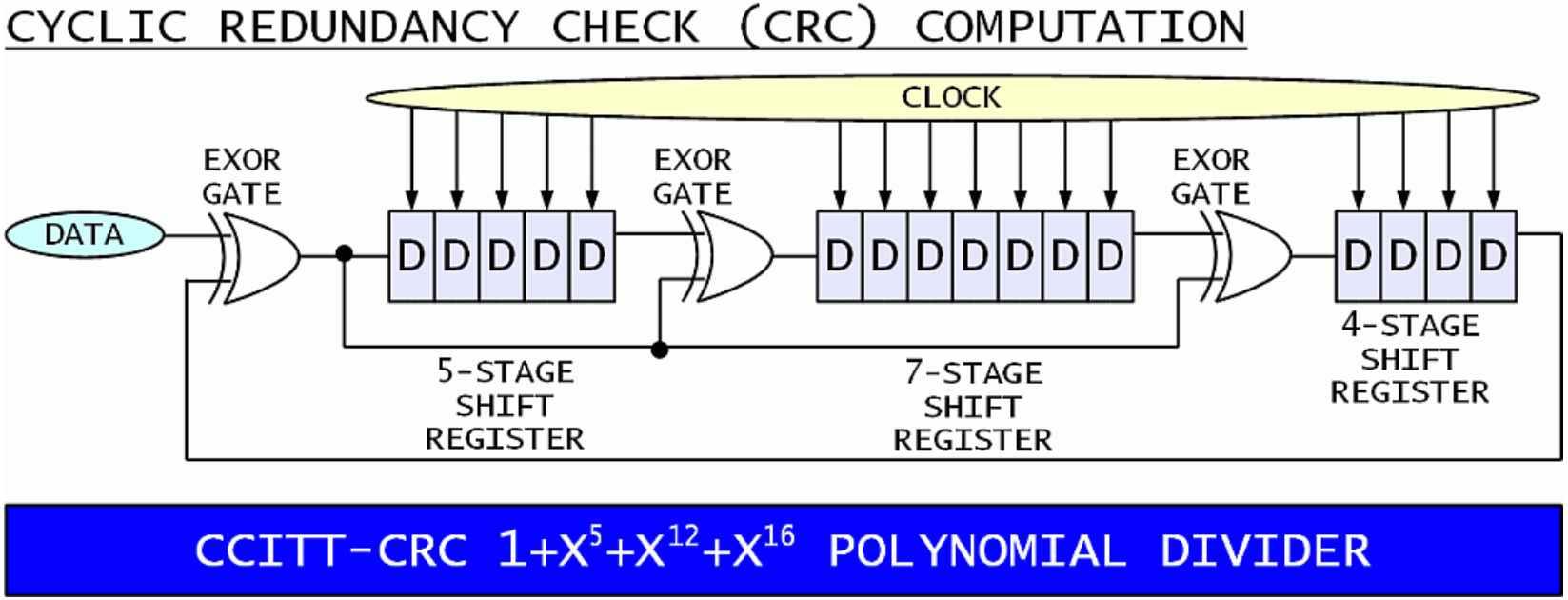
Поэтому

$D(m) \cdot x^k + mod(D(m) \cdot x^k, G) = \underline{Q \cdot G}$ будет делиться на G **без остатка.**

Свойства методов контроля по CRC

- Функция $mod(X, G)$ обладает свойством $mod(A \oplus B, G) = mod(A, G) \oplus mod(B, G)$
- Искажения при передаче кодового слова D можно представить как кодовое слово ошибок E: $D' = D \oplus E$, поэтому, если D это правильное кодовое слово, то $mod(D, G) = 0$ и $mod(D', G) = mod(E, G)$
- Оптимальный образующий полином G не должен раскладываться на произведение полиномов меньшей степени (должен быть неприводимым)
- CRC код с оптимальным образующим полиномом G степени k способен обнаруживать одиночные пакеты ошибок длиной до k битов
- Метод контроля по паритету является простейшим случаем CRC с полиномом 1 степени: $x^1 + 1$
- Примеры распространенных образующих полиномов:
  - CRC-4: x^4+x^1+1
  - CRC-16: $x^{16}+x^{15}+x^2+1$ (IBM), $x^{16}+x^{12}+x^5+1$ (CCITT)
  - CRC-32: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$
  - CRC-64: …….

Реализация CRC



Метод расчета CRC реализуется несложной аппаратной схемой состоящей из:

- регистров сдвига (shift register)
- логических элементов «исключающее или» (exor gate)

Схема получает принимаемое кодовое слово бит за битом от старшего к младшему. После последнего бита в сдвиговом регистре получается остаток

Другие методы контроля

Семейство **кодов Рида-Соломона** (Reed–Solomon codes, RS-codes) - 1969

- использует кодовое слово ***(n,m)*** **байтов** (типично ***(255,233)***)
- обнаруживает ошибки на уровне байтов (символов), а не битов
- обнаруживающая способность ***n – m*** байтов, исправляет до ***(n – m)/2*** байтов, если позиции ошибочных байтов неизвестны и до ***(n – m)*** байтов, если позиции ошибочных байтов определены (выпавшие символы)
- широко применяется при хранении данных (CD, DVD), в 2D-штрихкодах, DVB-S, ADSL и др.
- двухуровневая схема кодирования CIRC: RS(32,28) -> interleave -> RS(255,251)
- двухуровневая схема кодирования DVD: RS(208,192) -> intrlv -> RS(182,172)

Семейство кодов с проверкой паритета низкой плотности (**LDPC codes**) - 1993

- практически вплотную приближает скорость передачи к теоретическому пределу, обусловленному уровнем помех
- применяется: DVB-S2/T2/C2, 10Gbit Ethernet, G.hn, Wi-Fi 802.11n (option)

Семейство турбо-кодов (**Turbo codes**)

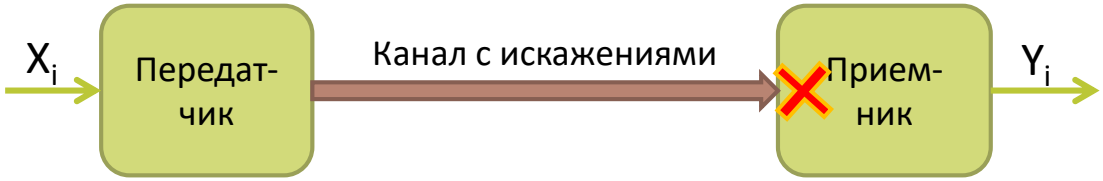
- эффективность сравнима с LDPC
- применяются в 3G/4G сетях сотовой связи, WiMAX, NASA (Mars Missions)

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - 6.1. Функция кадрирования
    - 6.2. Методы контроля правильности передачи данных
    - **6.3. Восстановление данных, искаженных в процессе передачи**
    - 6.4. Управление звеном передачи данных
    - 6.5. Семейство протоколов HDLC
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

Восстановление ошибочных данных

На приеме зафиксировано искажение переданных данных. Что делать дальше ?



Применение **FEC (forward error correction)** – помехоустойчивого кодирования.

Передатчик, используя ECC передает в канал данные с избытком, Благодаря этому приемник имеет возможность самостоятельно восстановить частично искаженные (потерянные данные). Пример:

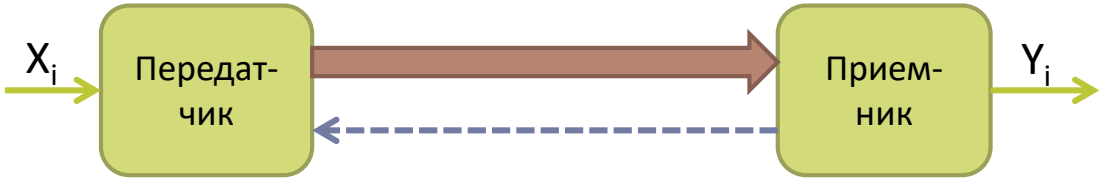
Исходные данные (числа) ***a,b,c*** по каналу передаются в виде ***a,b,c,d (d=a+b+c).*** Одно из чисел лишнее (избыточное, не несет полезной информации), зато оно позволяет восстановить данные в случае искажения любого одного из чисел ***a,b,c,d*** в процессе передачи.

Протоколы **ARQ (automatic repeat query)** – автоматического запроса повторения.

В этом случае реализуется повторная передача ошибочных данных

Принцип работы ARQ

ARQ – это протокол, т.е. алгоритм в котором участвуют и приемник и передатчик



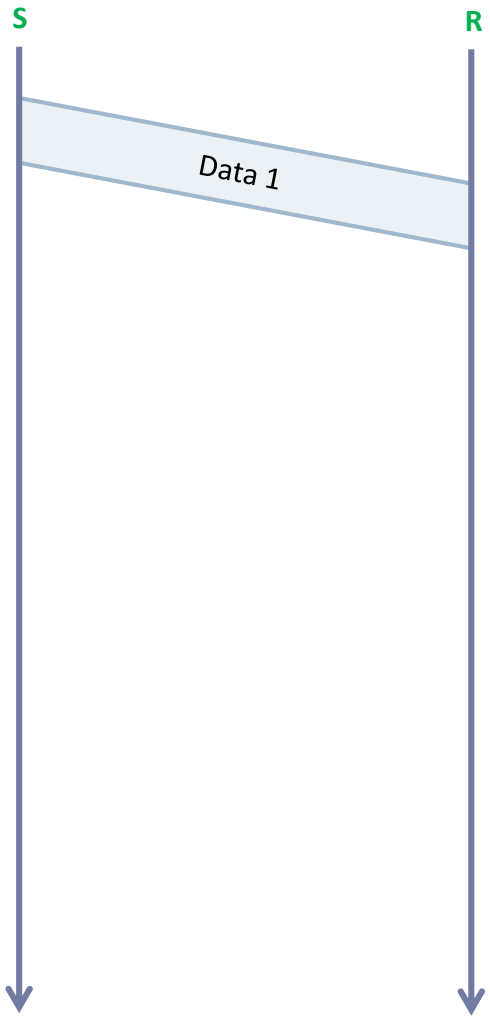
Для ARQ требуется канал обратной связи (feedback channel) от приемника к передатчику:

- В направлении Передатчик→Приемник передается последовательность пронумерованных блоков данных (кадров/сегментов…)
- В направлении Приемник→Передатчик передаются ARQ-сигналы – примитивные сообщения ARQ протокола

Сигналами приемник информирует передатчик о том, какие кадры он принял, а передатчик организует повторную передачу не принятых кадров. Классы ARQ:

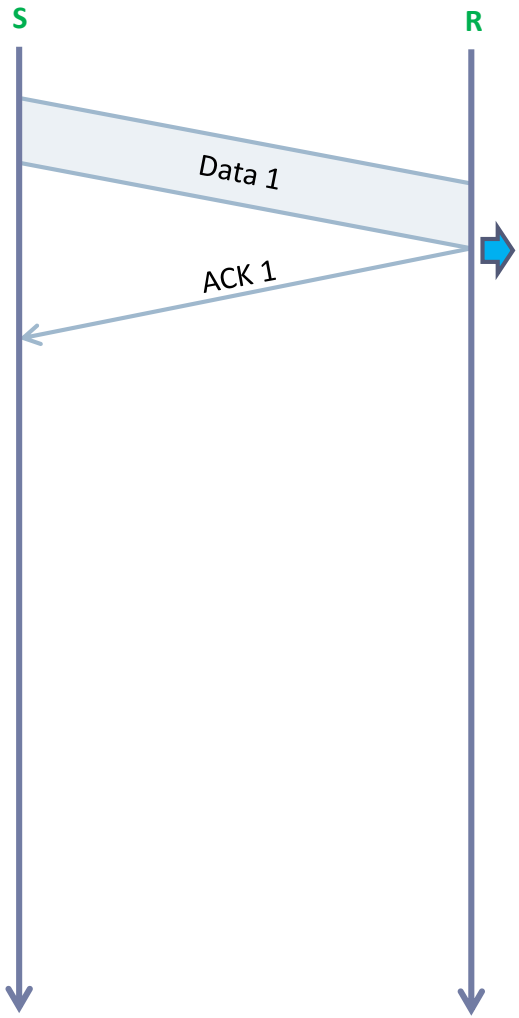
- Простой ARQ с остановкой и ожиданием (stop-and-wait ARQ)
- ARQ с возвращением на N кадров (go-back-N ARQ)
- ARQ с выборочным повтором (selective repeat ARQ)

Простой ARQ с остановкой и ожиданием



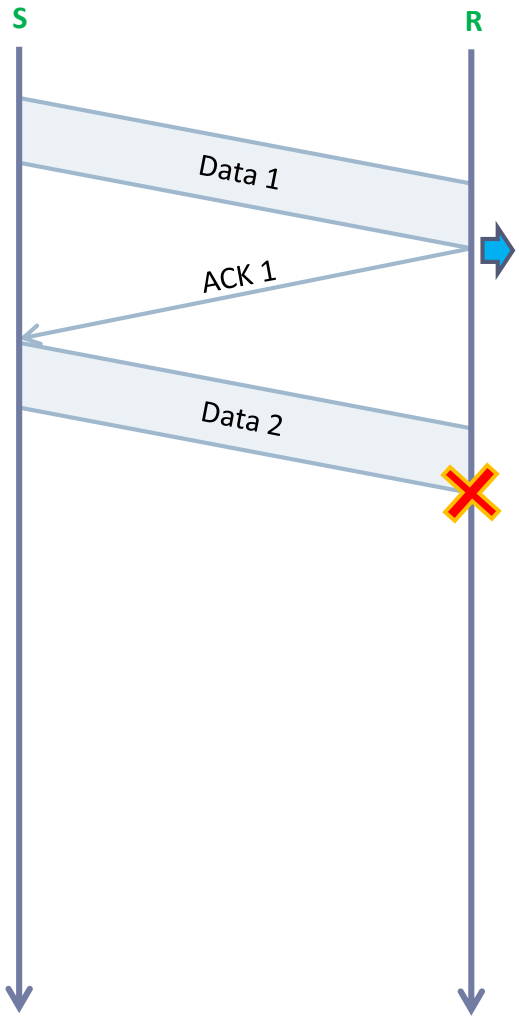
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)

Простой ARQ с остановкой и ожиданием



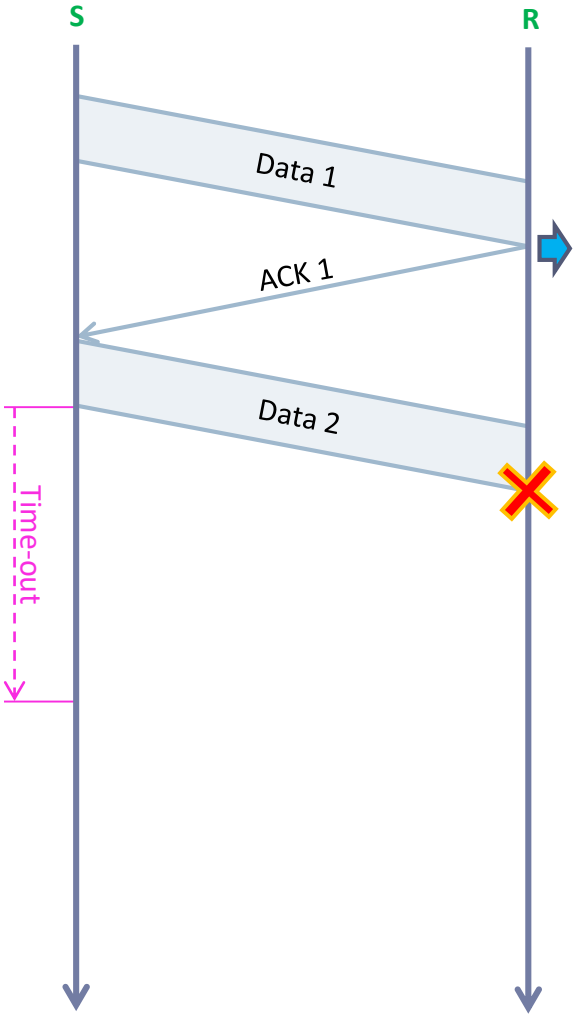
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)
- Получив кадр *приемник* направляет ACK (acknowledgement – подтверждение)

Простой ARQ с остановкой и ожиданием



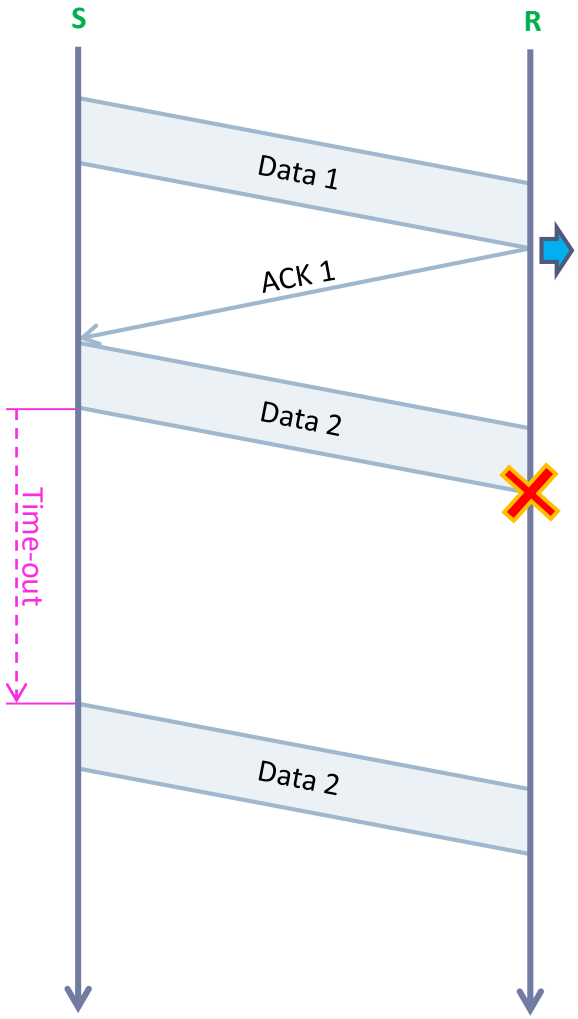
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)
- Получив кадр *приемник* направляет ACK (acknowledgement – подтверждение)
- Только получив ACK на кадр ***1*** *передатчик* получает право передать кадр ***2***
- Любой из узлов обязан игнорировать кадры, принятые с ошибкой (несоответствием CRC)

Простой ARQ с остановкой и ожиданием



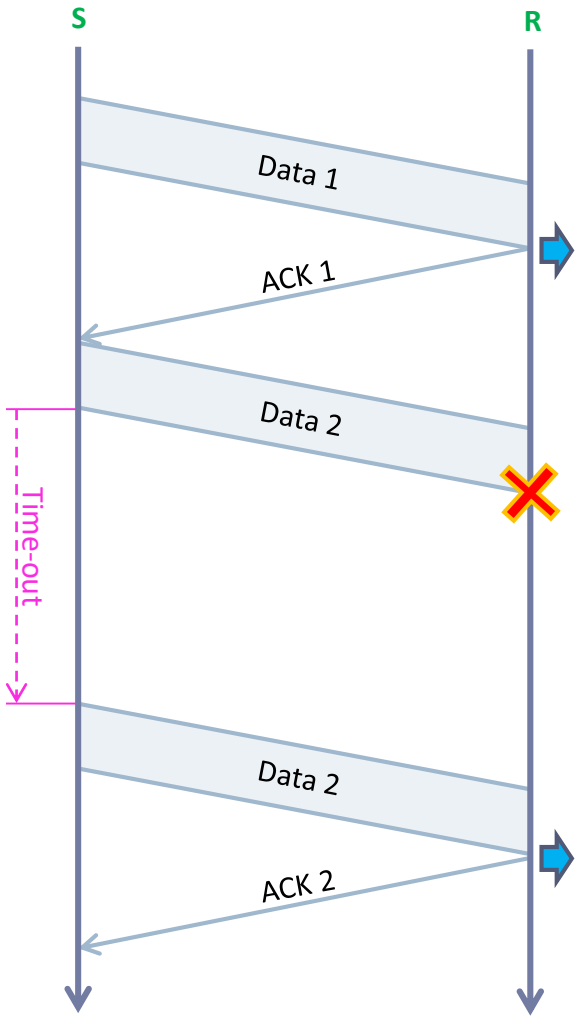
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)
- Получив кадр *приемник* направляет ACK (acknowledgement – подтверждение)
- Только получив ACK на кадр ***1*** *передатчик* получает право передать кадр ***2***
- Любой из узлов обязан игнорировать кадры, принятые с ошибкой (несоответствием CRC)
- Отсутствие ACK приводит к истечению времени ожидания ответа (таймауту) в *передатчике*

Простой ARQ с остановкой и ожиданием



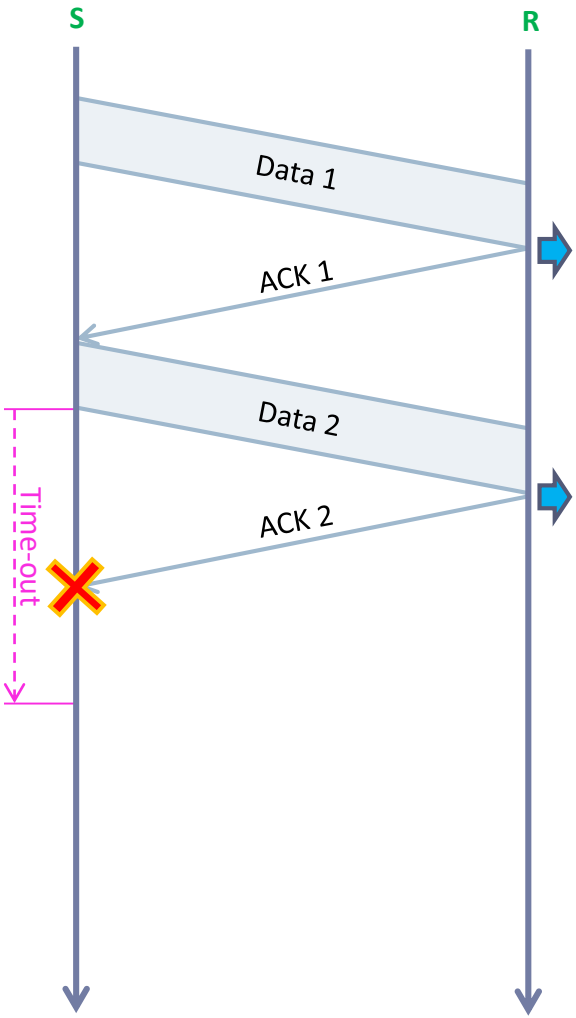
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)
- Получив кадр *приемник* направляет ACK (acknowledgement – подтверждение)
- Только получив ACK на кадр ***1*** *передатчик* получает право передать кадр ***2***
- Любой из узлов обязан игнорировать кадры, принятые с ошибкой (несоответствием CRC)
- Отсутствие ACK приводит к истечению времени ожидания ответа (таймауту) в *передатчике*
- При возникновении таймаута *передатчик* обязан повторно передать последний кадр

Простой ARQ с остановкой и ожиданием



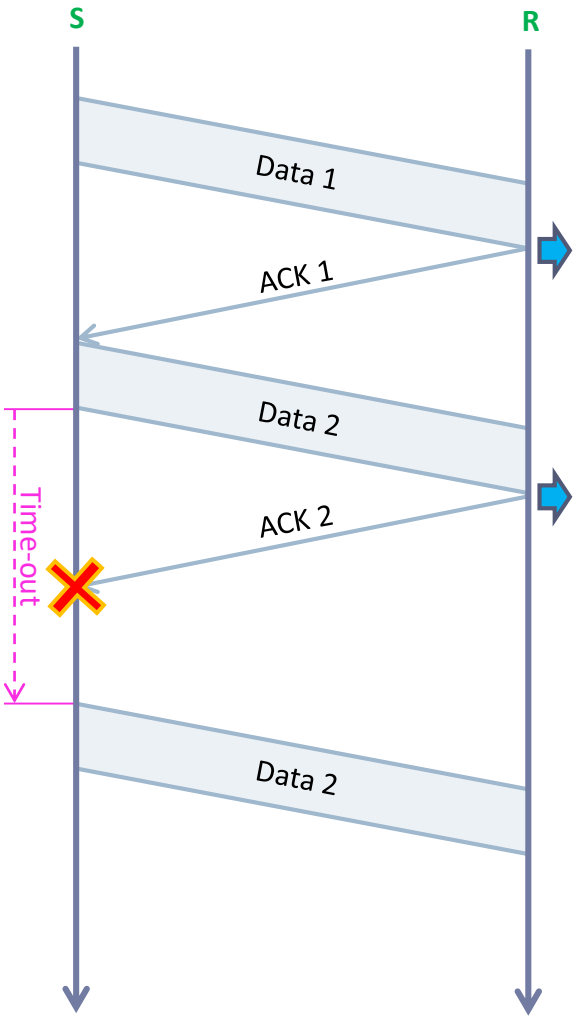
- Передав очередной кадр *передатчик* останавливается и ждет (stop and wait)
- Получив кадр *приемник* направляет ACK (acknowledgement – подтверждение)
- Только получив ACK на кадр ***1*** *передатчик* получает право передать кадр ***2***
- Любой из узлов обязан игнорировать кадры, принятые с ошибкой (несоответствием CRC)
- Отсутствие ACK приводит к истечению времени ожидания ответа (таймауту) в *передатчике*
- При возникновении таймаута *передатчик* обязан повторно передать последний кадр
- Получив корректный кадр *приемник* подтверждает его обычным порядком

Простой ARQ с остановкой и ожиданием



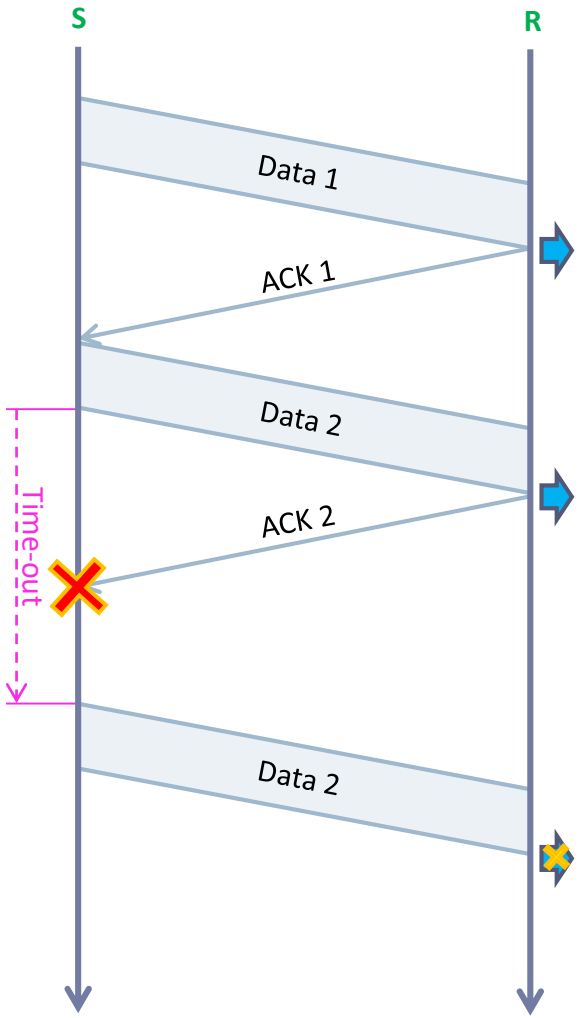
- Причиной возникновения таймаута может быть также и искажение самого кадра ACK (*передатчик* тоже обязан игнорировать кадр, принятый с ошибкой !)

Простой ARQ с остановкой и ожиданием



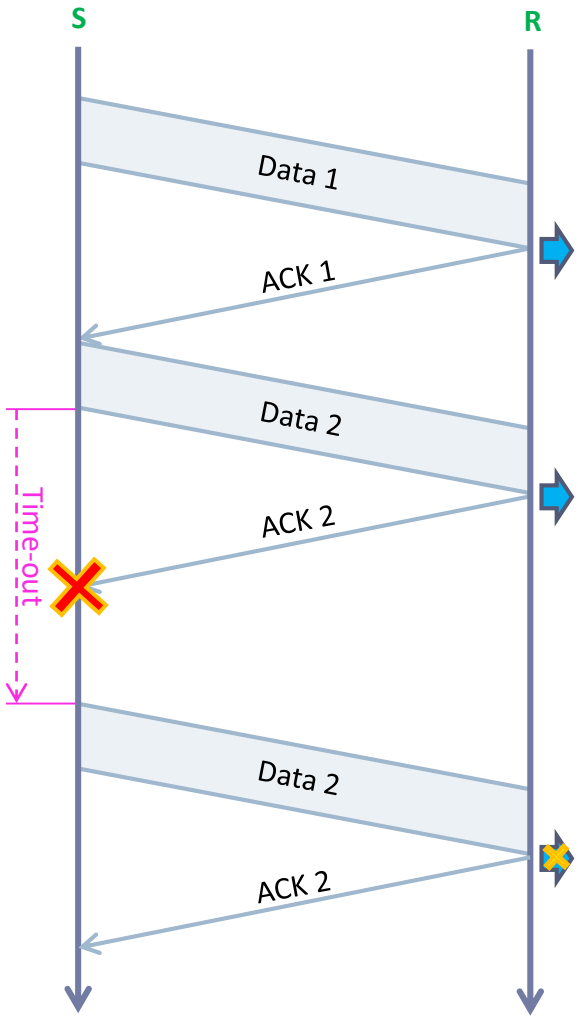
- Причиной возникновения таймаута может быть также и искажение самого кадра ACK (*передатчик* тоже обязан игнорировать кадр, принятый с ошибкой !)
- *Передатчик* должен повторно передать кадр 2, который уже принят *приемником*

Простой ARQ с остановкой и ожиданием



- Причиной возникновения таймаута может быть также и искажение самого кадра ACK (*передатчик* тоже обязан игнорировать кадр, принятый с ошибкой !)
- *Передатчик* должен повторно передать кадр 2, который уже принят *приемником*
- В этом случае приемник получит две копии одного кадра (*Data2*)
- Чтобы исключить размножение данных приемник обязан сравнивать номера предыдущего и теку-щего принятого кадров

Простой ARQ с остановкой и ожиданием



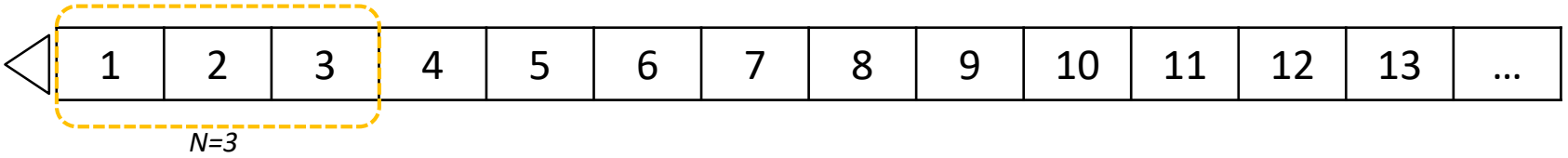
- Причиной возникновения таймаута может быть также и искажение самого кадра ACK (*передатчик* тоже обязан игнорировать кадр, принятый с ошибкой !)
- *Передатчик* должен повторно передать кадр 2, который уже принят *приемником*
- В этом случае приемник получит две копии одного кадра (*Data2*)
- Чтобы исключить размножение данных приемник обязан сравнивать номера предыдущего и теку-щего принятого кадров
- если они совпадают, то приемник игнорирует дубль данных, но все равно обязан послать ACK

Свойства и недостатки простого ARQ

- ✓ Перед началом передачи кадров *приемник* и *передатчик* должны быть приведены во взаимно-согласованное состояние: номер ***n(r)*** последнего принятого *приемником* кадра должен быть на единицу меньше ***n(s)*** – номера очередного передаваемого кадра в *передатчике*. Это требует выполнения в рамках ARQ протокола процедуры установки соединения
- ✓ В каждый момент времени в состоянии передачи (с момента начала отправки и до получения подтверждения ***ACK***) может находиться максимум один кадр
- ✓ Процедура восстановления ошибочно принятых данных инициируется только по истечению таймаута → существенные потери времени на восстановление
- ✓ Величина таймаута должна выбираться больше, чем максимально возможное значение RTT (round-trip time – время туда-обратно): время за которое короткий кадр (***ACK***) способен дойти от передатчика к приемнику и обратно
- ✓ Даже при отсутствии ошибок протокол простого ARQ с остановкой и ожиданием не способен полностью утилизировать ресурсы канала: при ожидании канал простаивает !!!

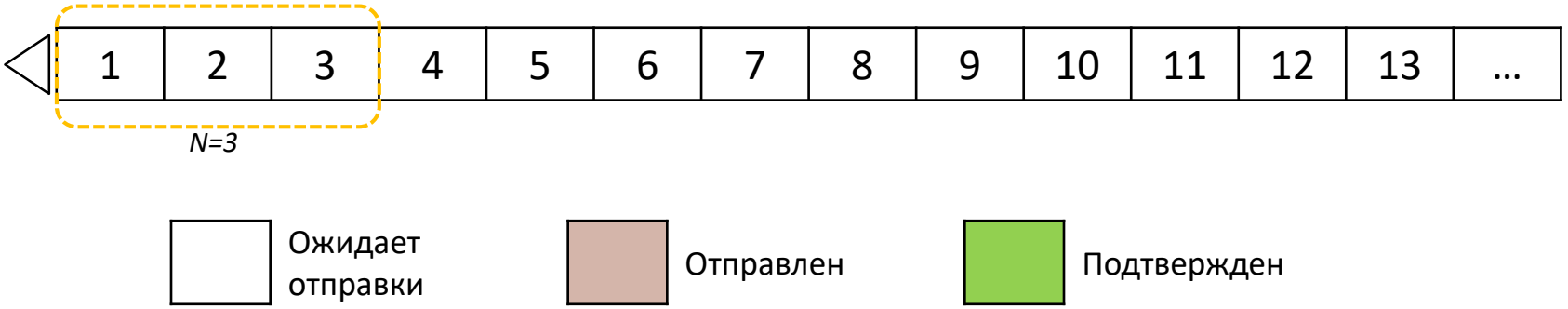
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



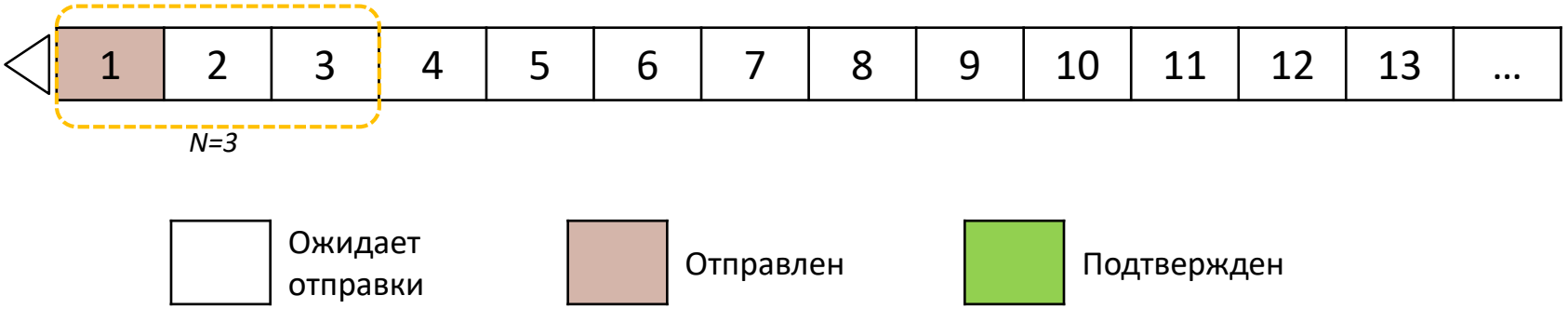
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



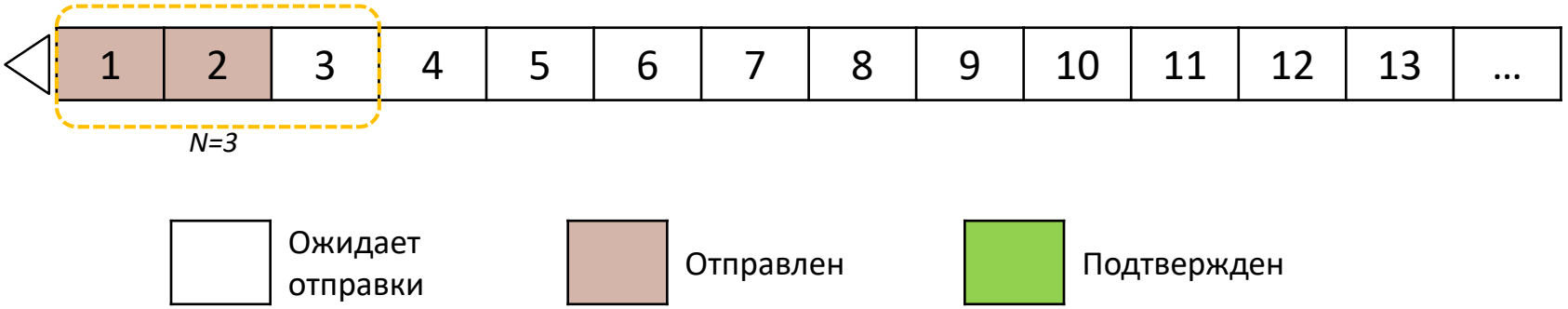
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



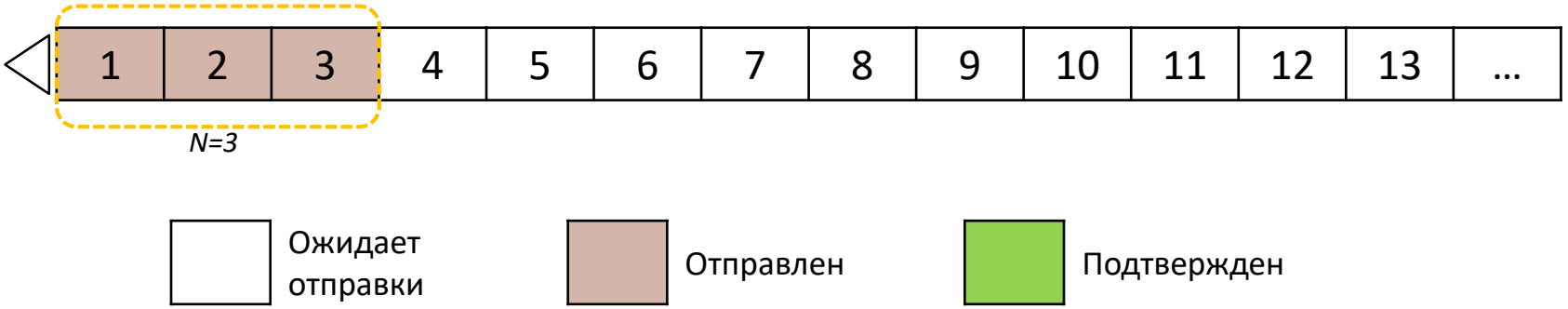
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



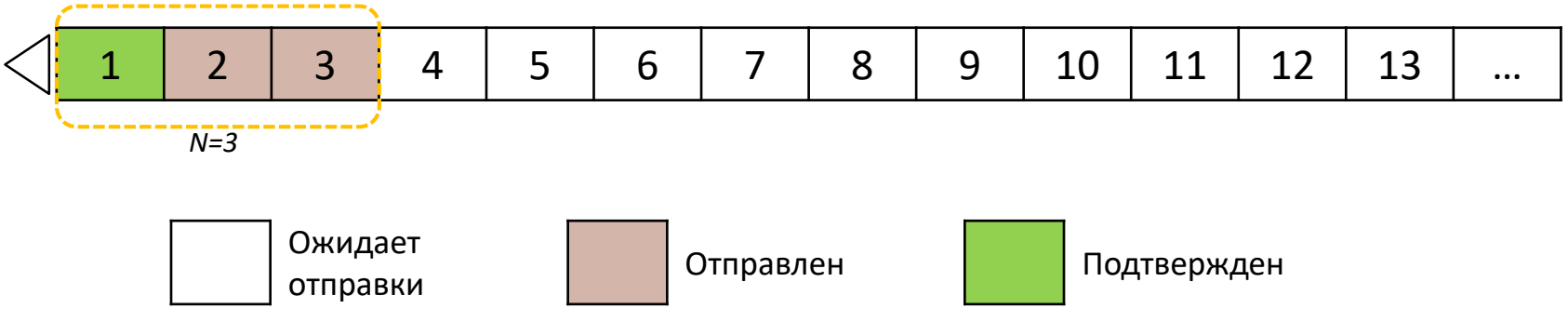
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



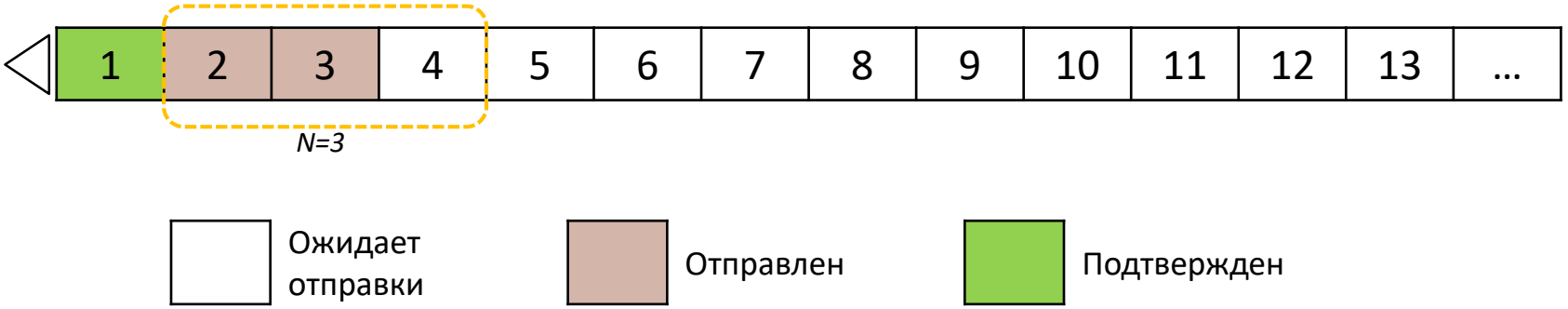
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



Принцип «скользящего окна»

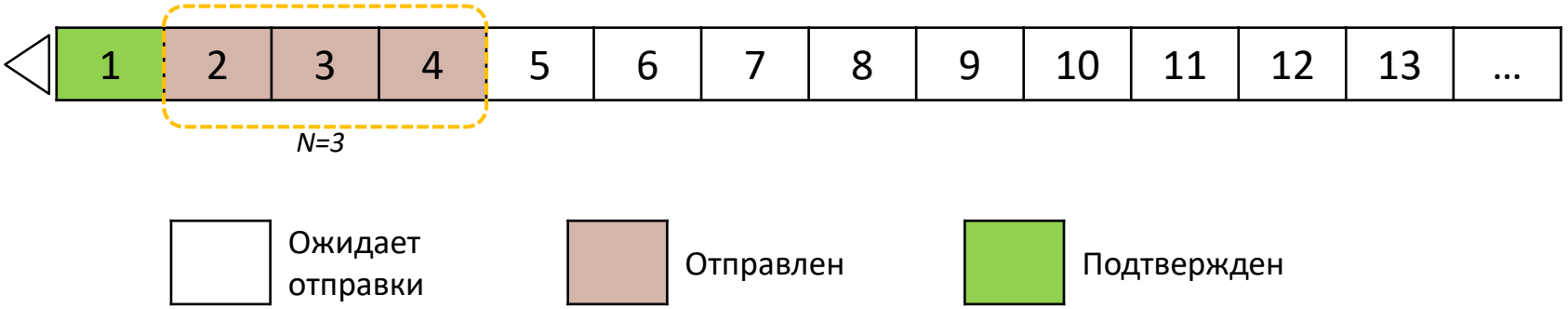
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.

Принцип «скользящего окна»

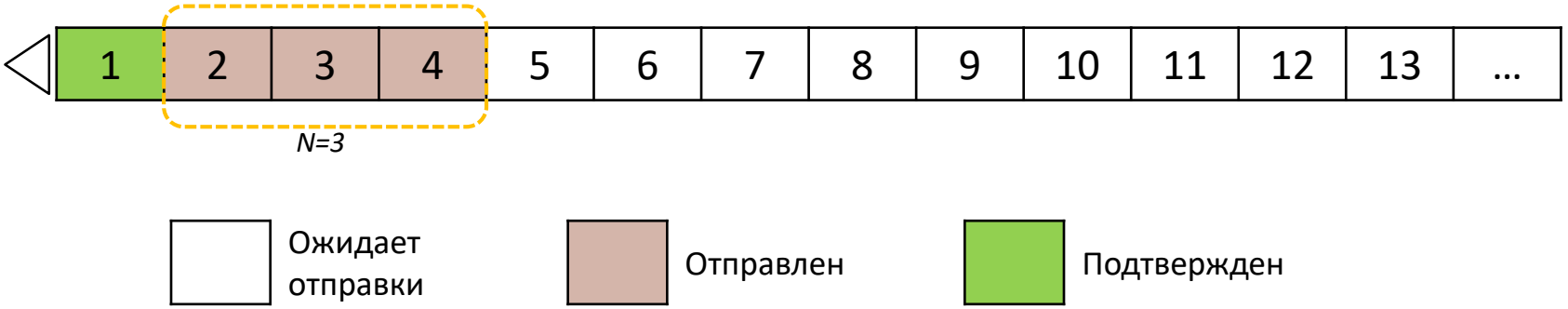
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***. Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.

Принцип «скользящего окна»

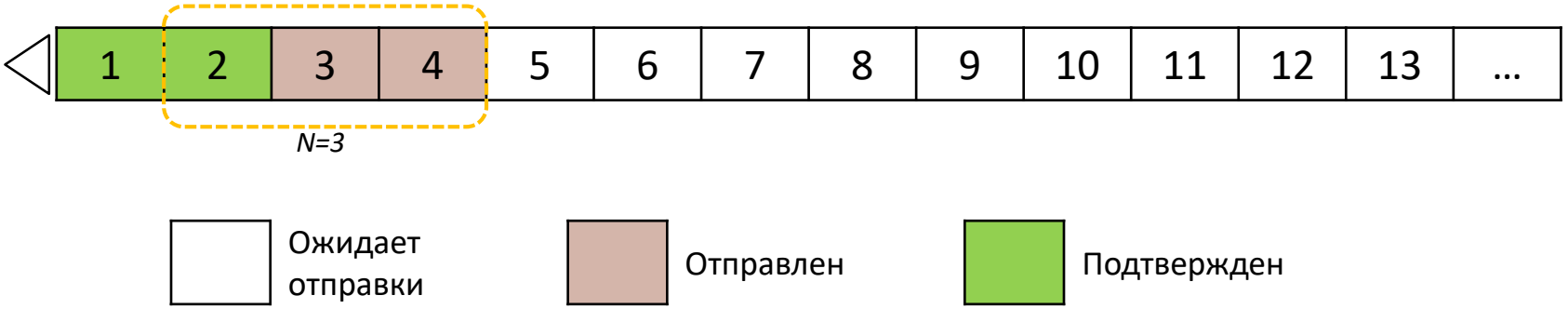
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

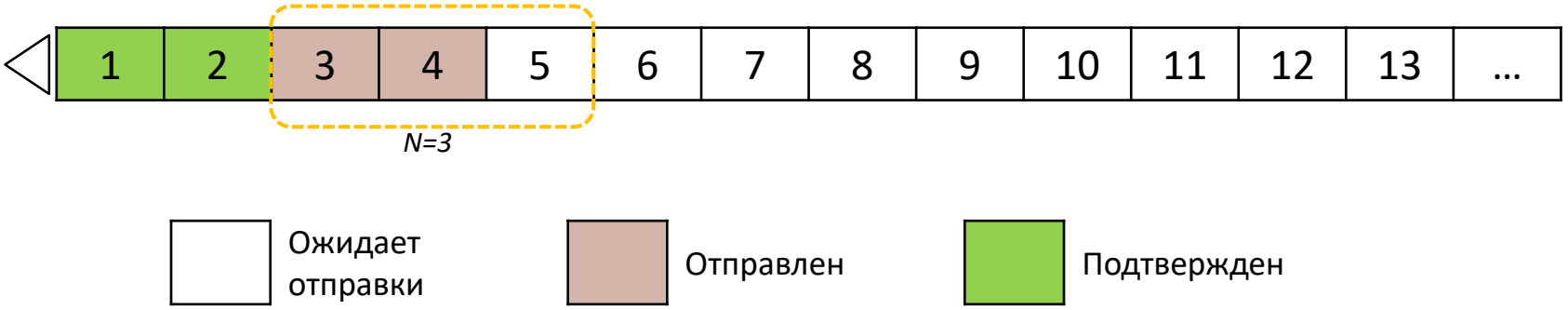
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

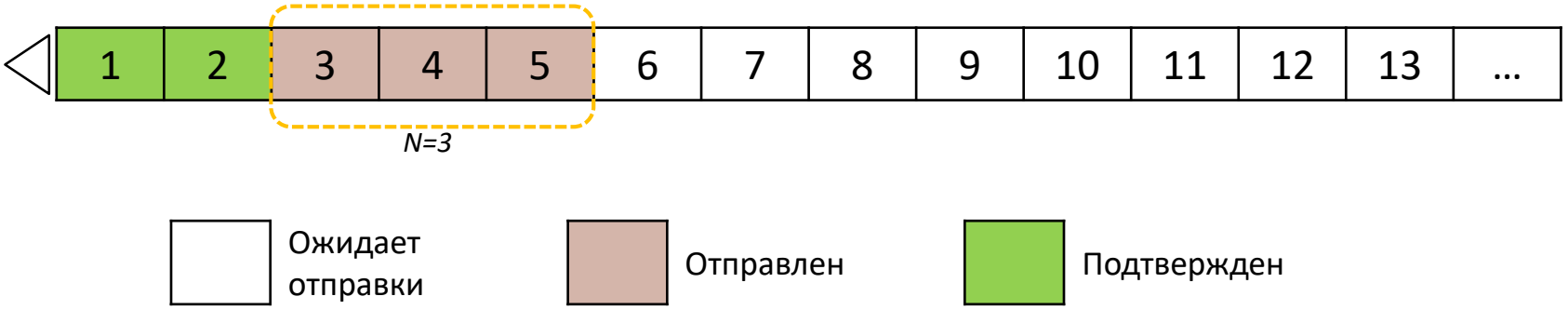
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

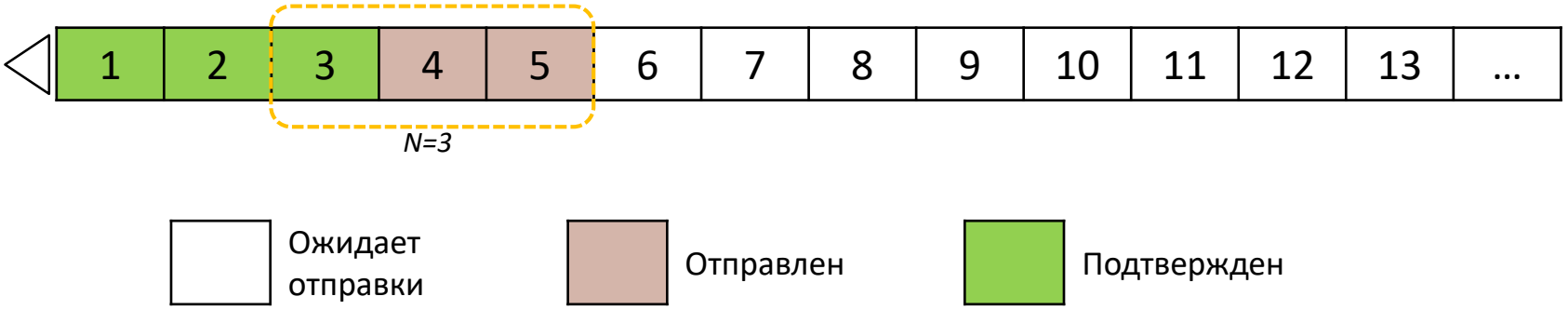
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

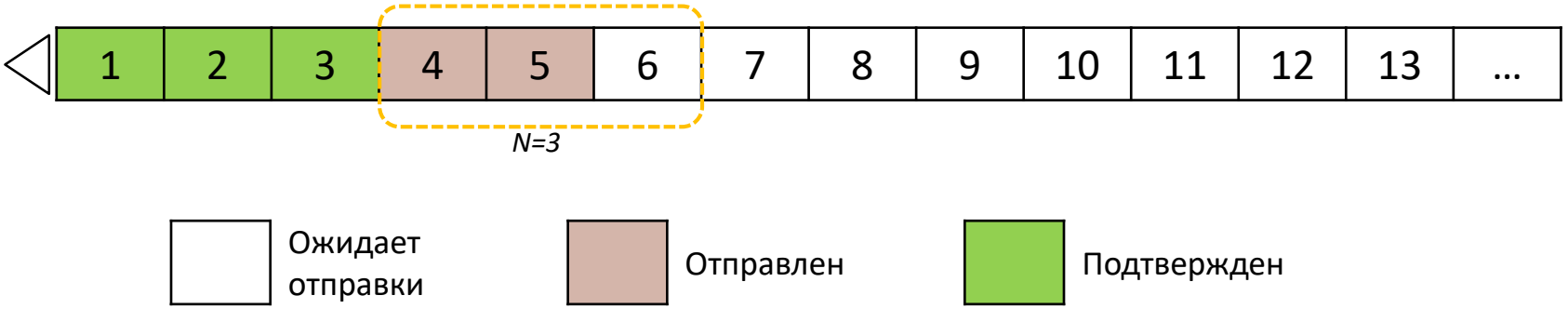
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

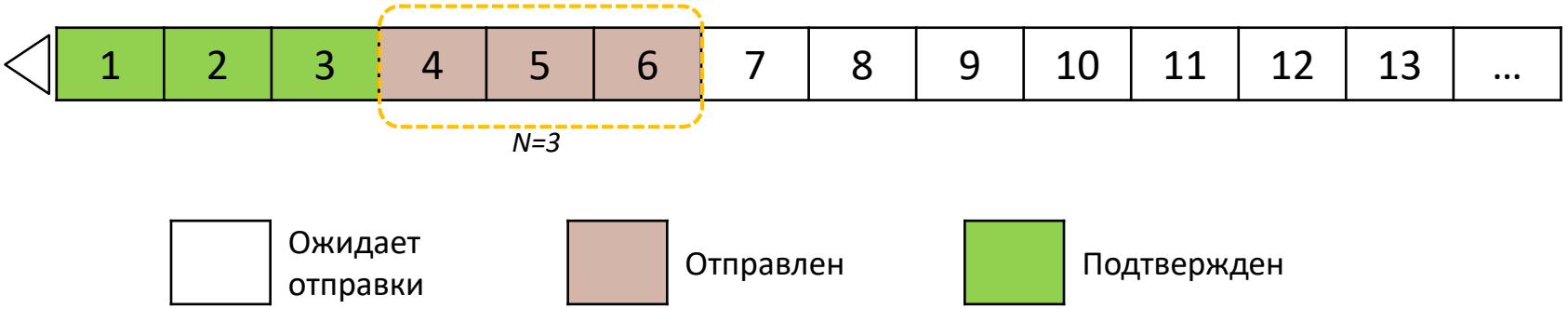
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

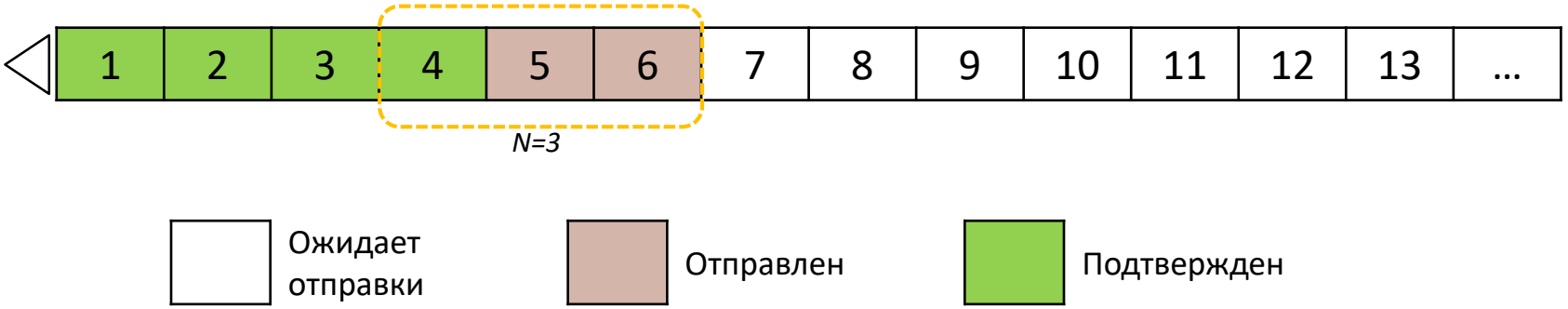
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

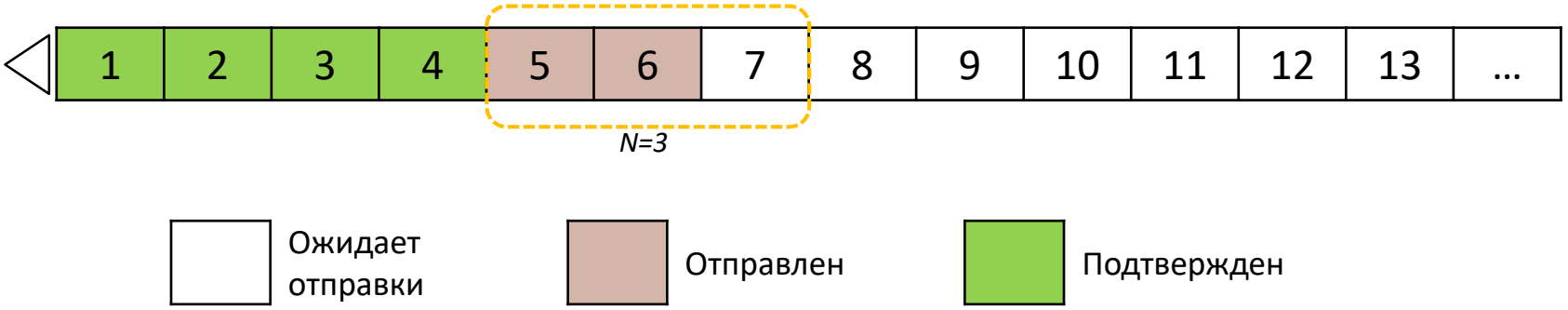
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

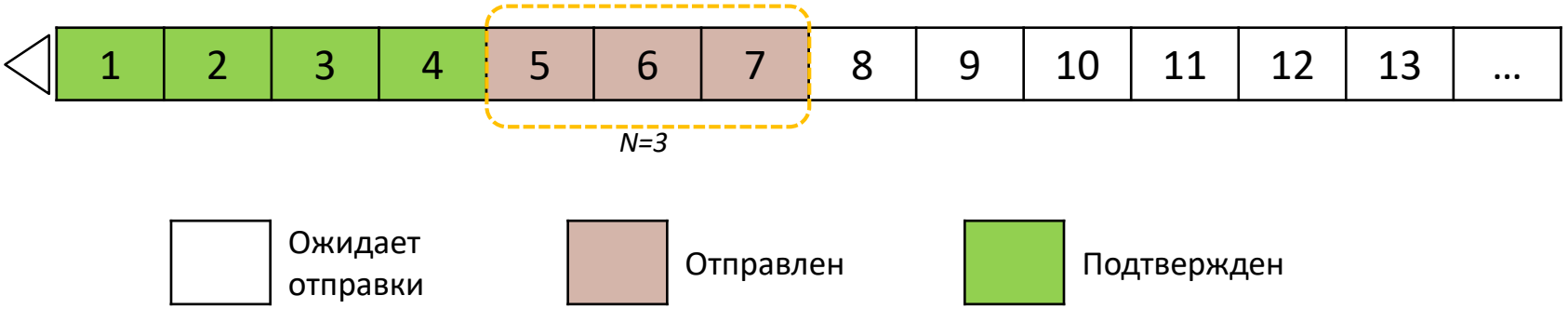
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

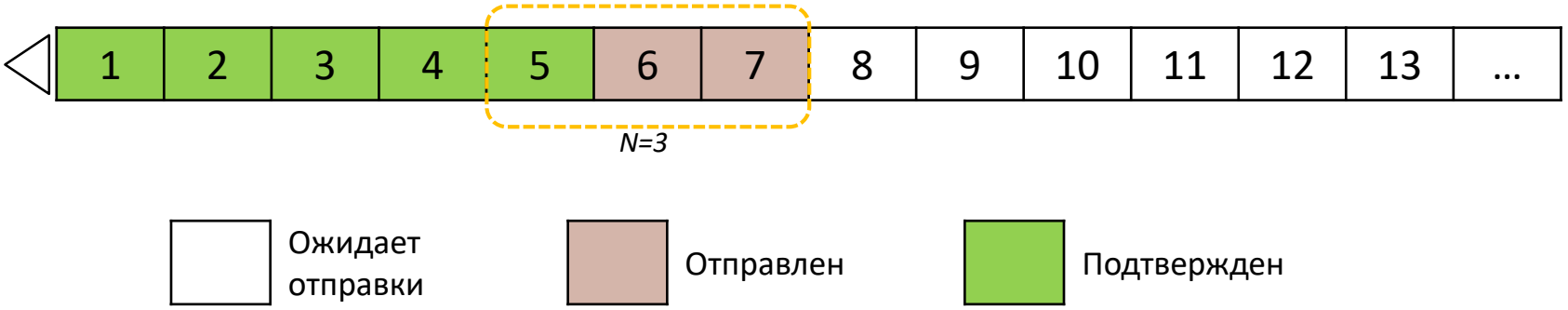
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

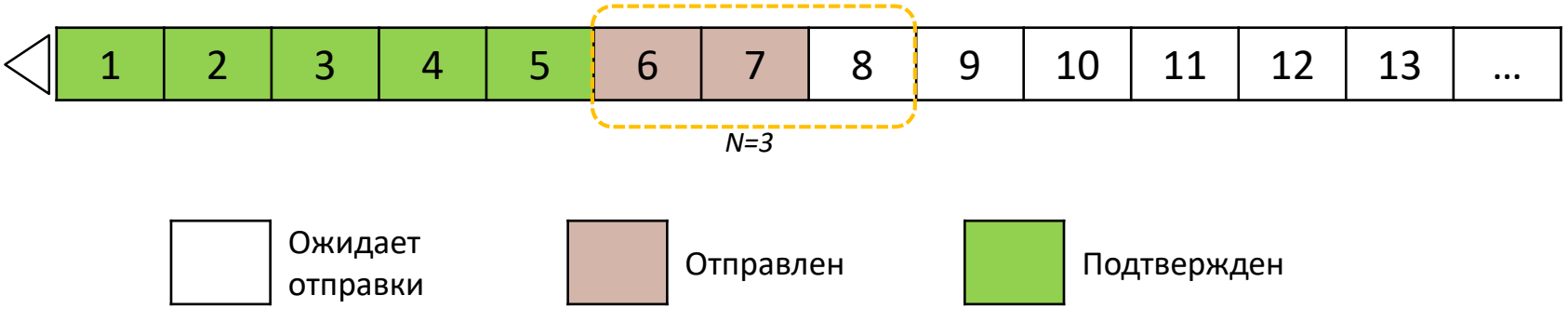
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

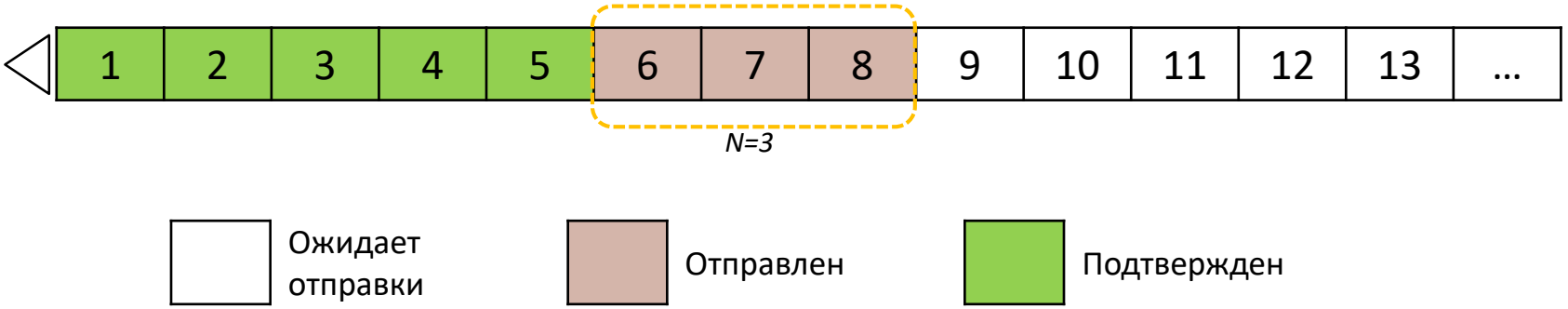
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

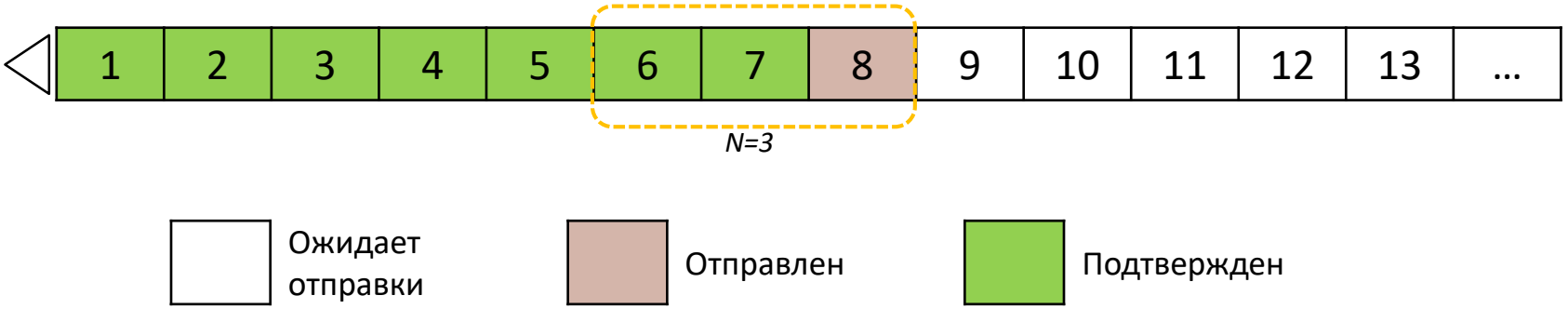
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

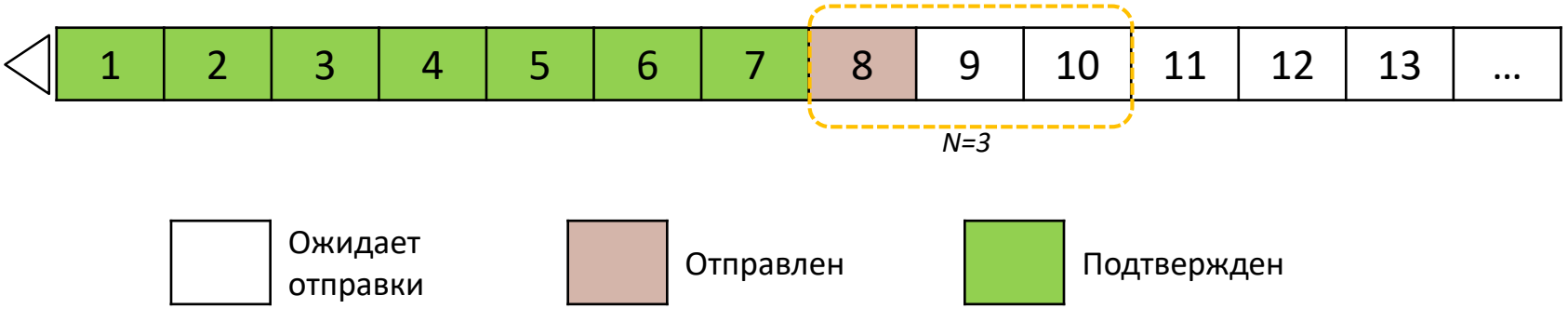
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

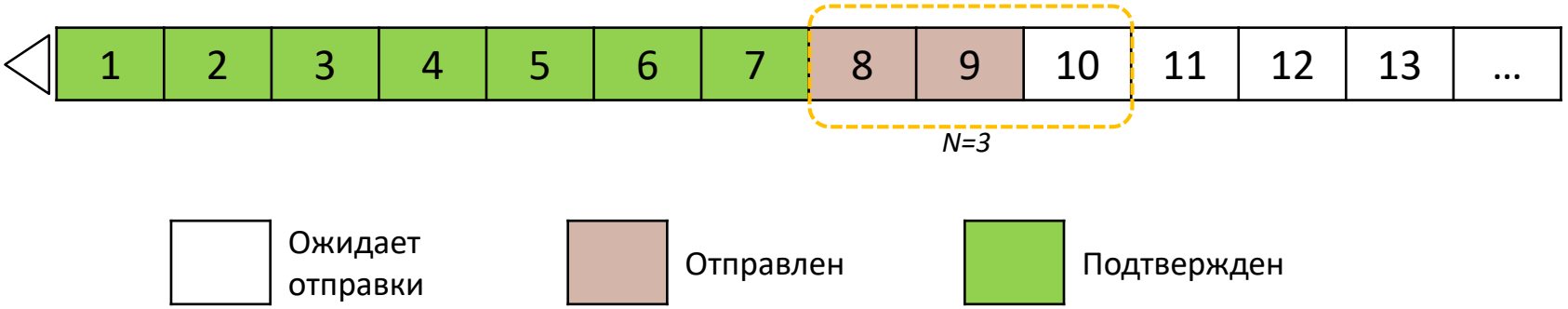
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

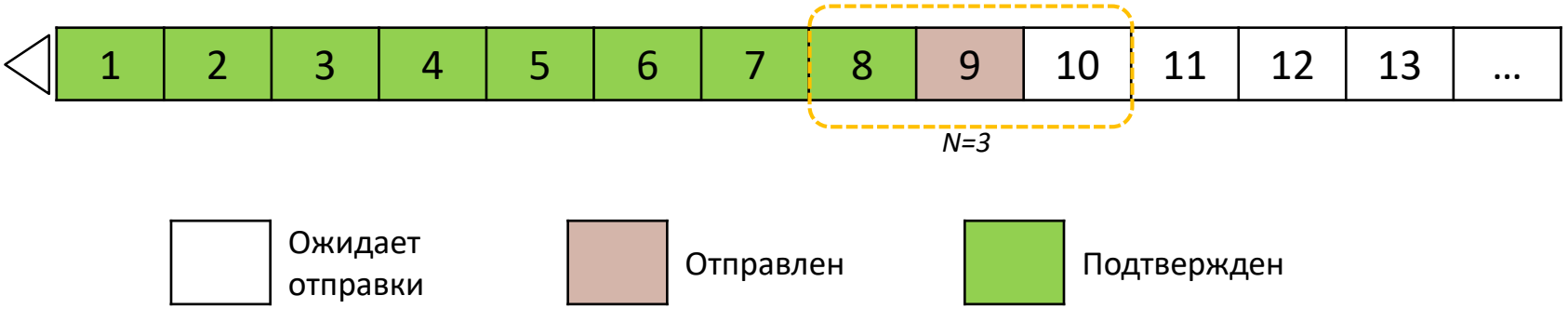
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

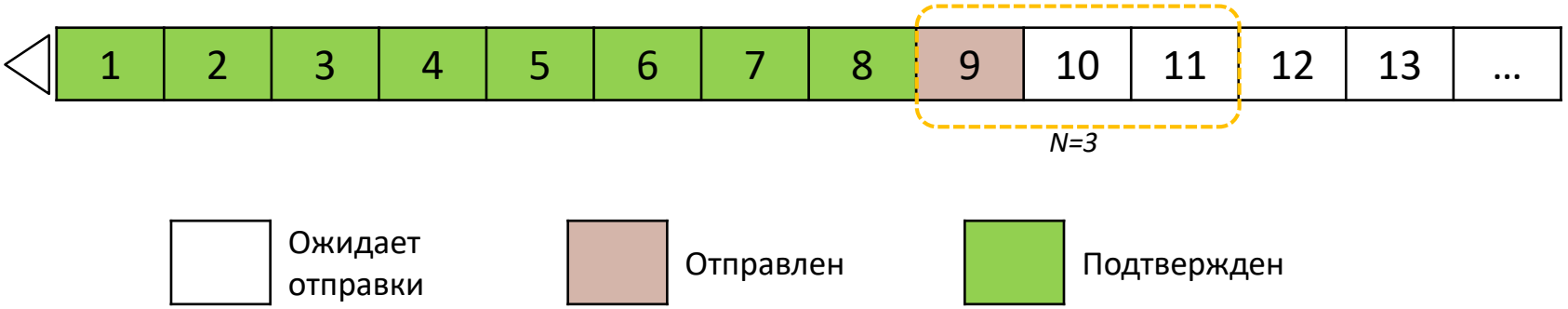
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

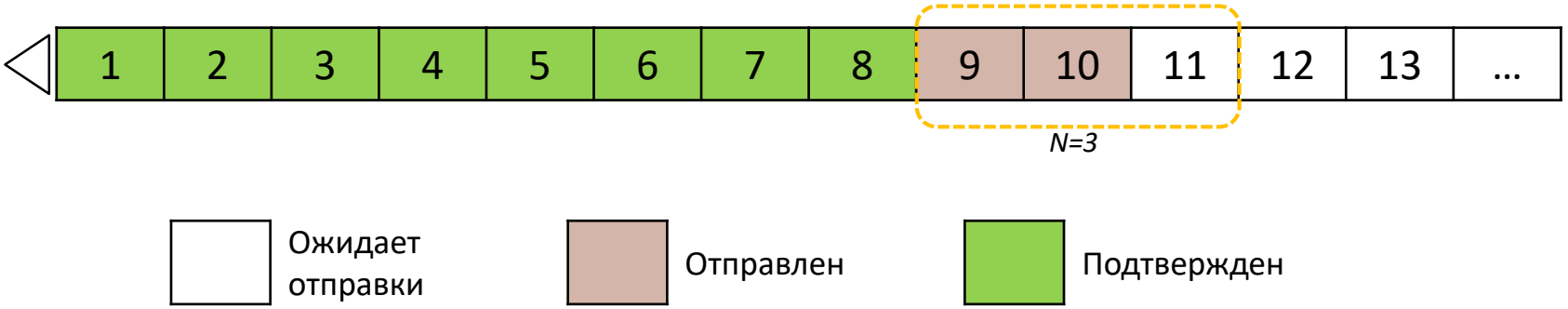
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

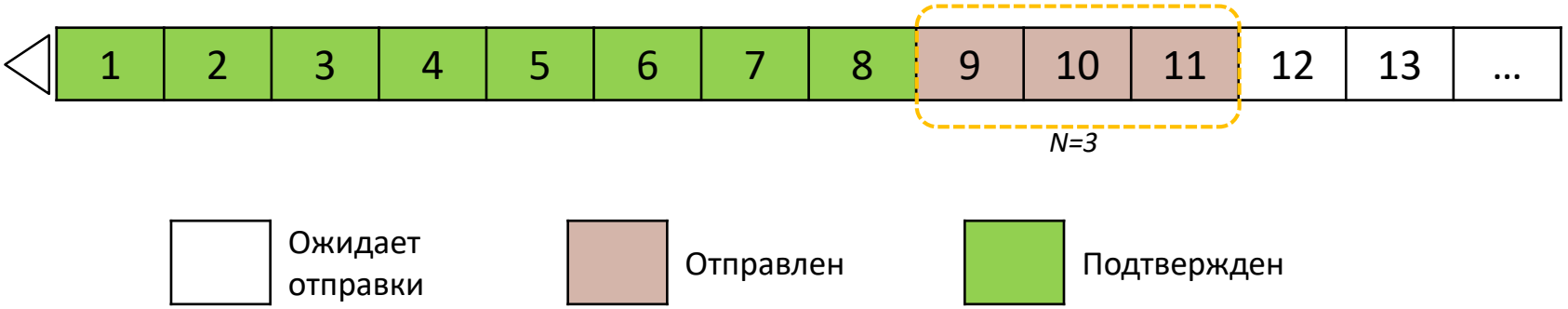
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

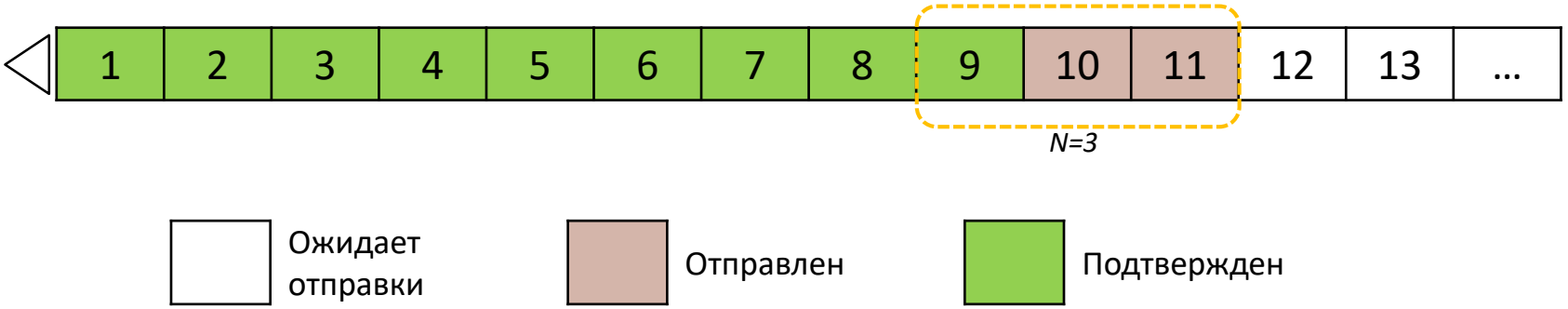
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

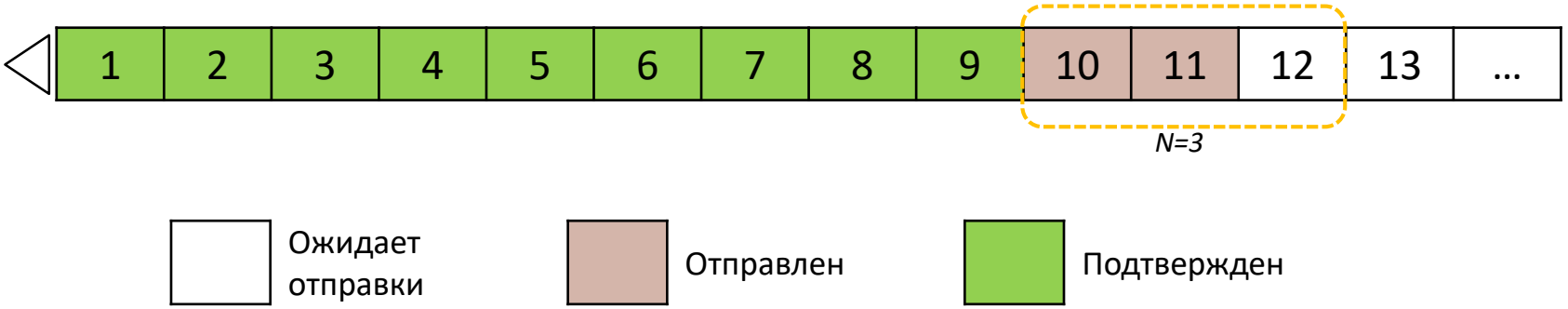
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

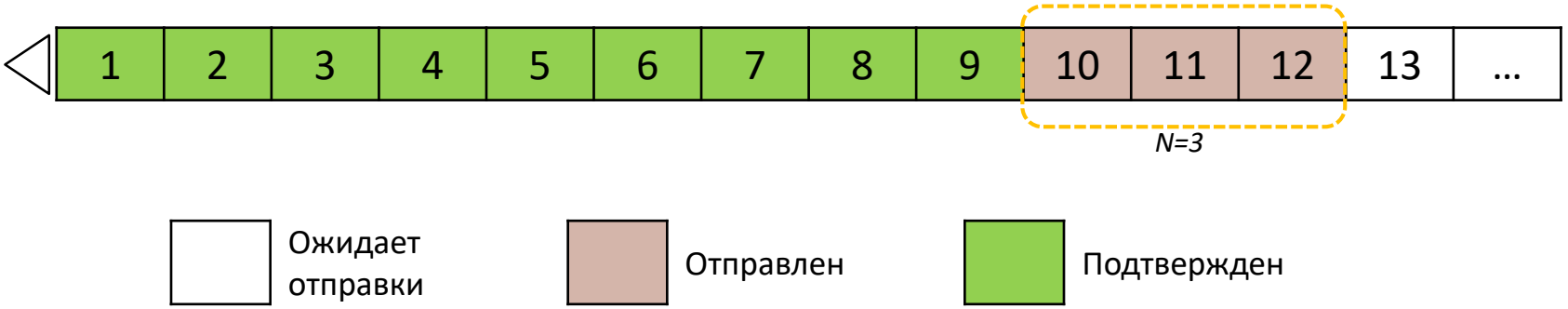
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (***i***-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

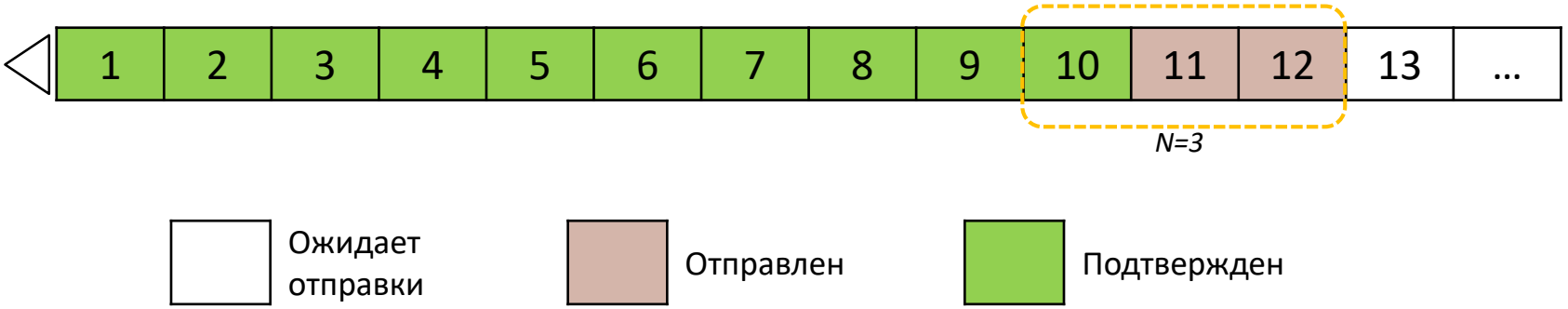
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

Принцип «скользящего окна»

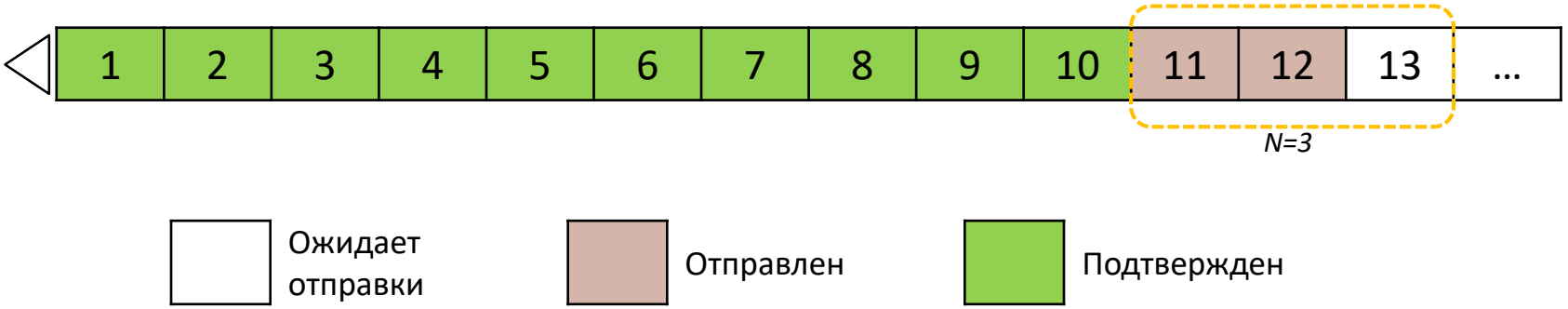
Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

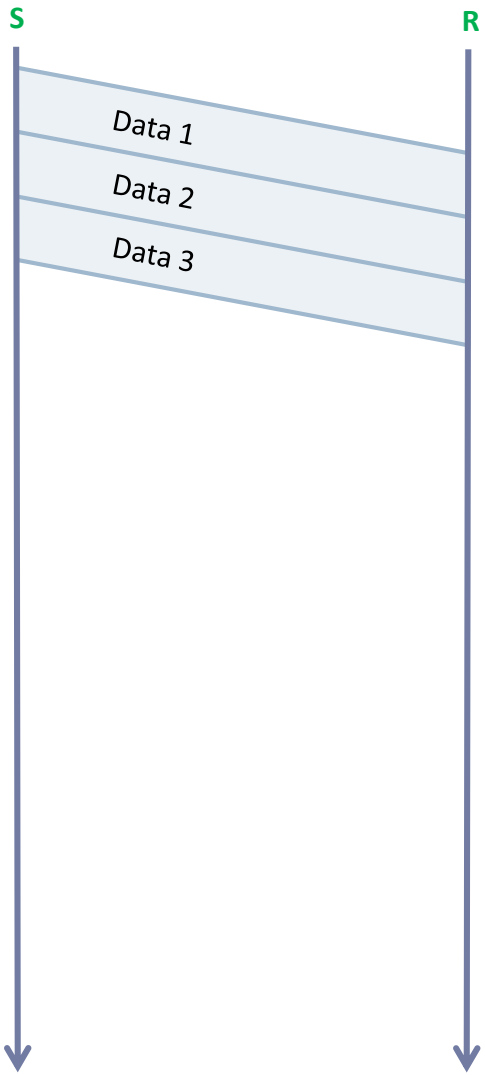
Принцип «скользящего окна»

Для исключения ожидания следует разрешить *передатчику* передавать кадры без остановки (не дожидаясь подтверждения), ограничив максимальное количество еще не подтвержденных кадров некоторой константой ***N***.
Совокупность еще не подтвержденных кадров от ***i+1*** до ***i+N*** (*i*-последний подтвержденный кадр) называется **окном передачи** (transmission window).



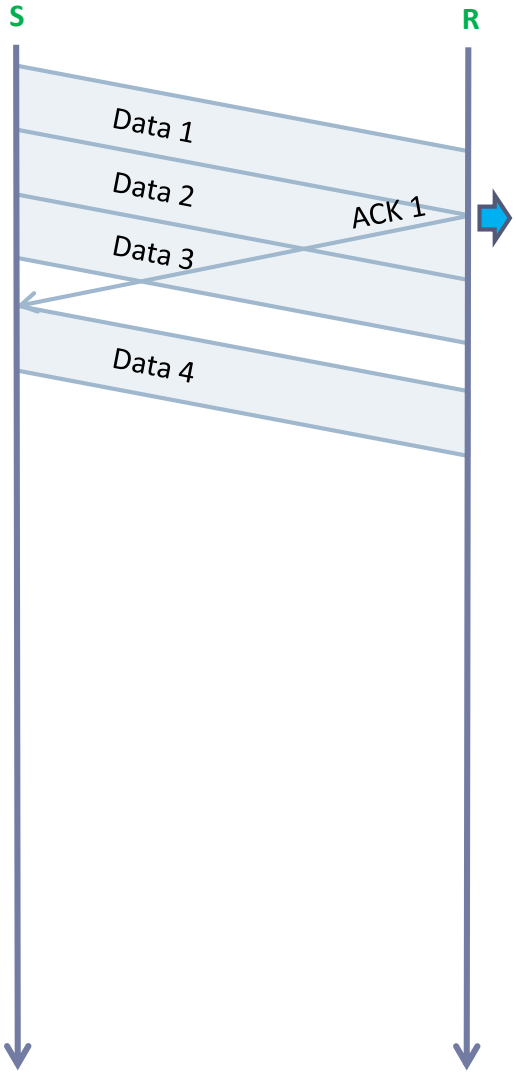
При получении подтверждения ***ACK(i)*** окно сдвигается (slides - «скользит») к позиции после кадра ***i***. Тем самым для передатчика открывается возможность передавать новые кадры.
Таким образом в каждый момент времени в состоянии передачи могут находится не более ***N*** кадров.

ARQ с возвращением на N кадров (Go-back-N)



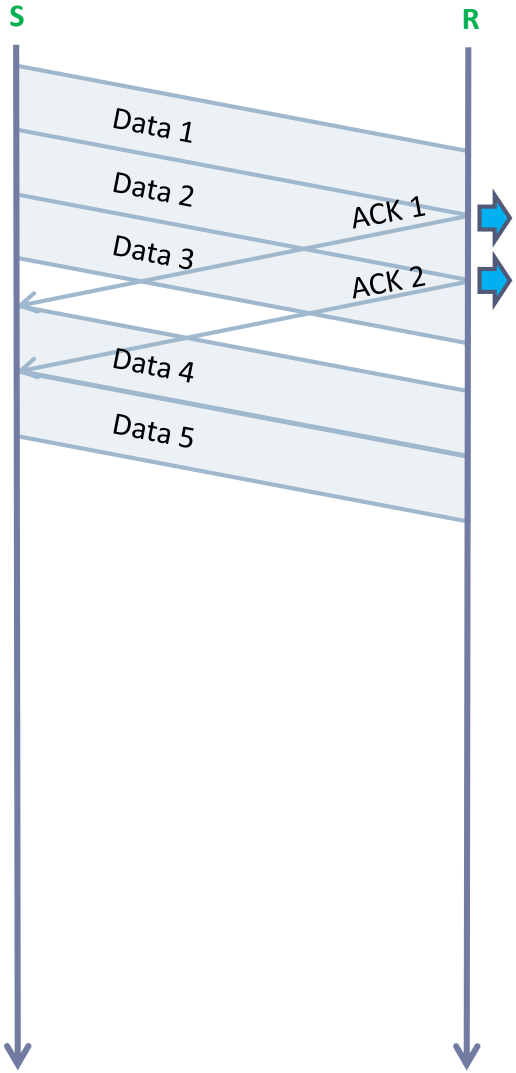
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)

ARQ с возвращением на N кадров (Go-back-N)



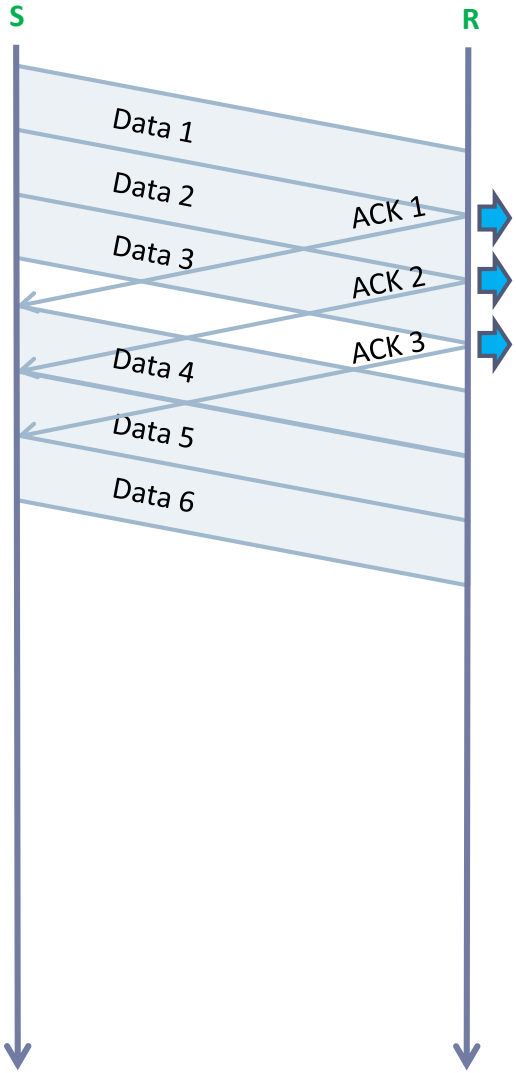
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4

ARQ с возвращением на N кадров (Go-back-N)



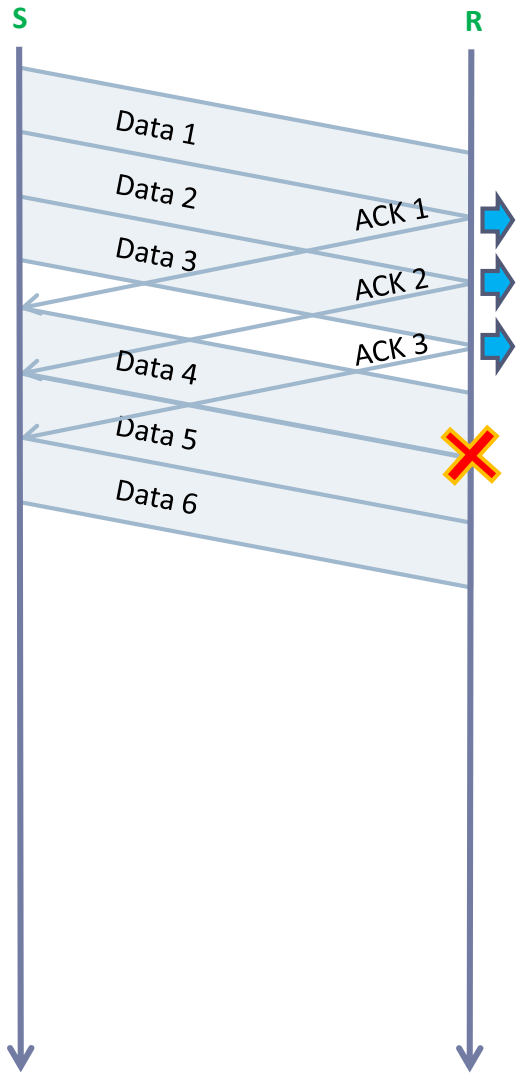
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5

ARQ с возвращением на N кадров (Go-back-N)



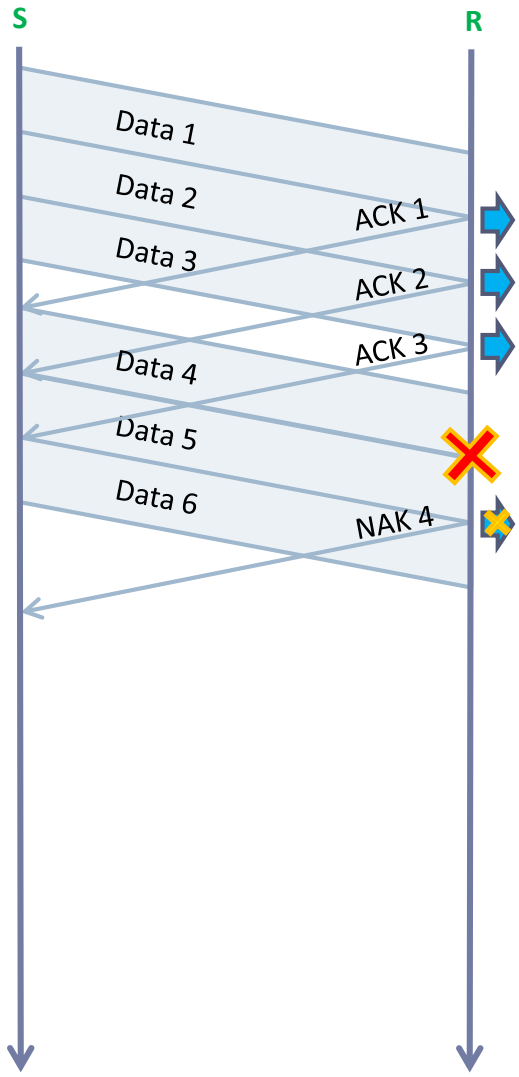
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6

ARQ с возвращением на N кадров (Go-back-N)



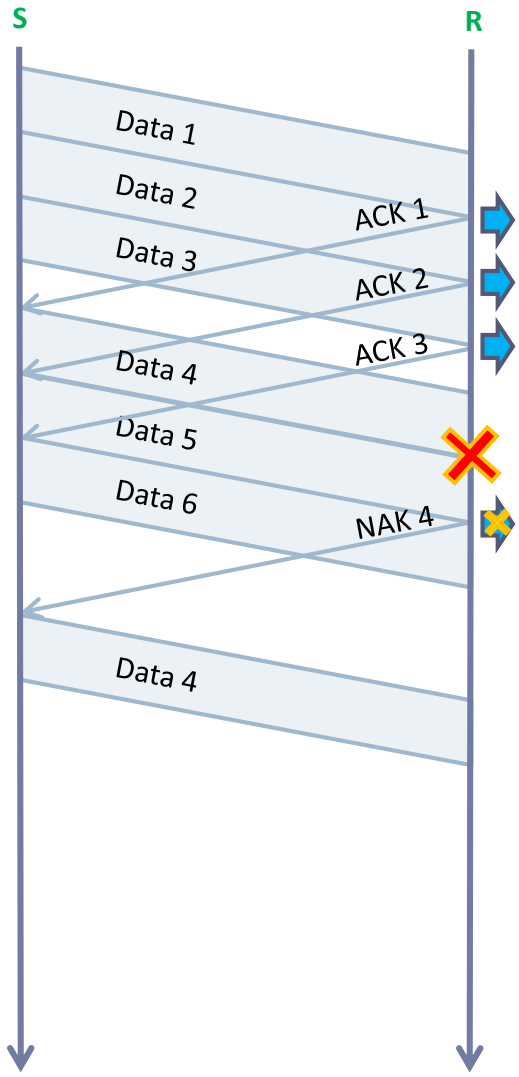
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*.

ARQ с возвращением на N кадров (Go-back-N)



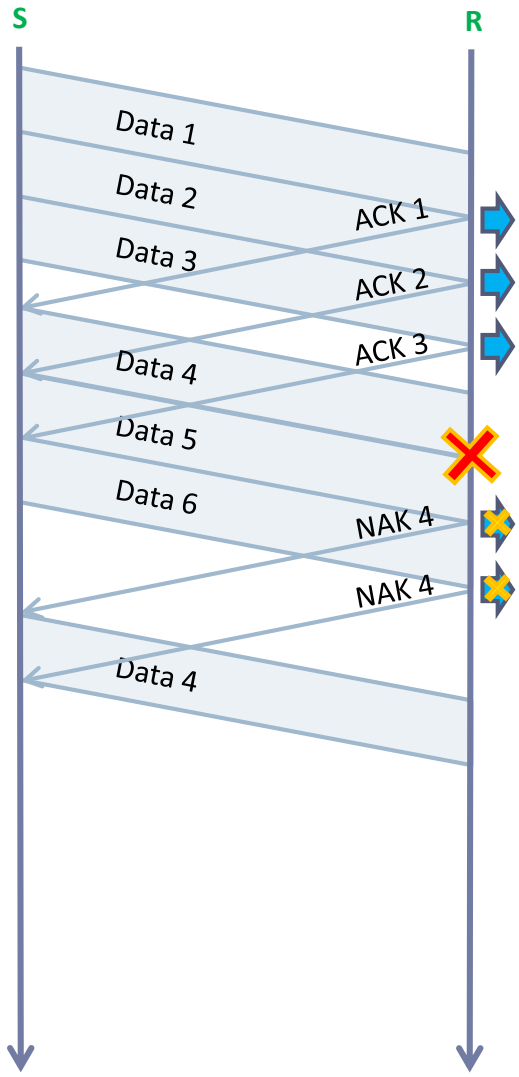
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)

ARQ с возвращением на N кадров (Go-back-N)



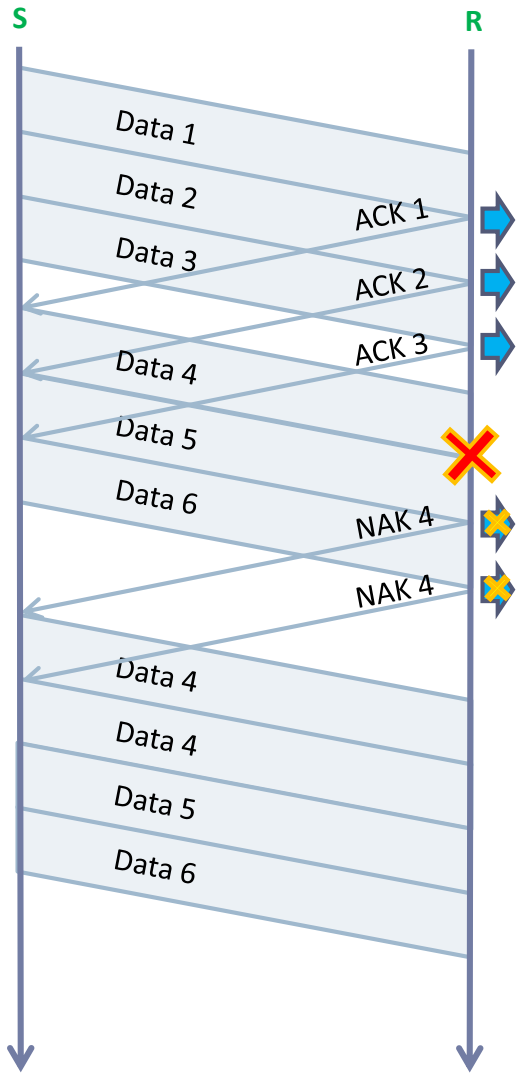
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)
- Получив ***NAK 4*** *передатчик* повторно передает кадры с 4 по 6, а затем продолжает сдвигать окно дальше в результате получения ACK 4, 5 и т.д.

ARQ с возвращением на N кадров (Go-back-N)



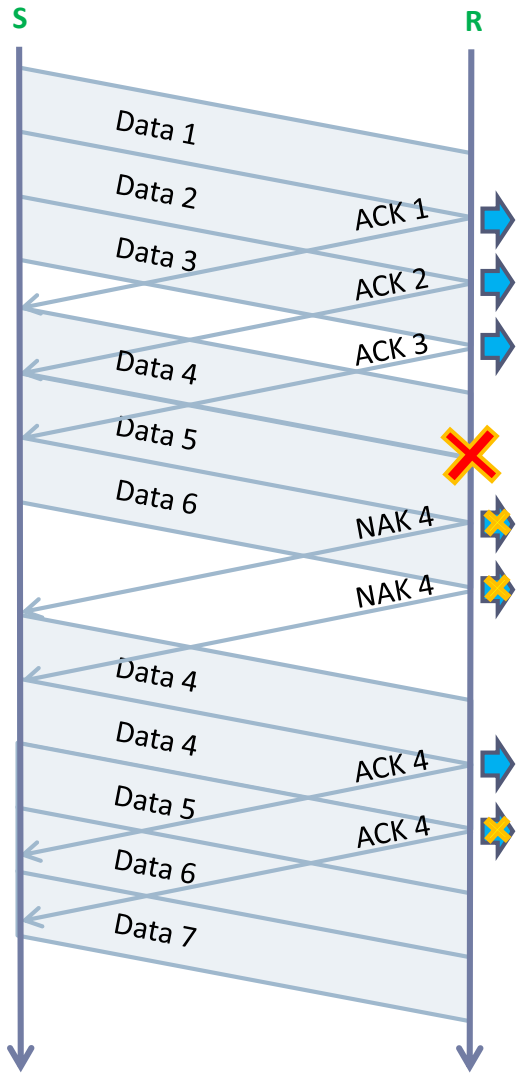
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)
- Получив ***NAK 4*** *передатчик* повторно передает кадры с 4 по 6, а затем продолжает сдвигать окно дальше в результате получения ACK 4, 5 и т.д.

ARQ с возвращением на N кадров (Go-back-N)



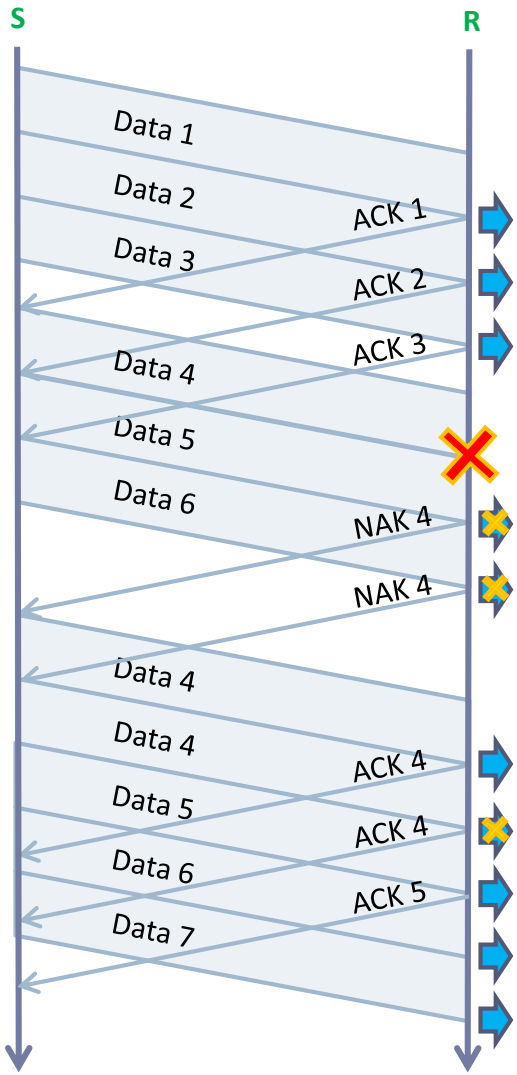
- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)
- Получив ***NAK 4*** *передатчик* повторно передает кадры с 4 по 6, а затем продолжает сдвигать окно дальше в результате получения ACK 4, 5 и т.д.

ARQ с возвращением на N кадров (Go-back-N)



- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)
- Получив ***NAK 4*** *передатчик* повторно передает кадры с 4 по 6, а затем продолжает сдвигать окно дальше в результате получения ACK 4, 5 и т.д.

ARQ с возвращением на N кадров (Go-back-N)



- Передав первый кадр *передатчик* продолжает передавать кадры в пределах текущего окна не дожидаясь подтверждения. Только когда окно полностью исчерпано *передатчик* останавливается и ждет получения подтверждения (если размер окна недостаточен)
- Получение ***ACK 1*** сдвигает окно в позицию 2-4, ***ACK 2*** – в позицию 3-5, ***ACK 3*** – в позицию 4-6. *Передатчик* отправляет кадры с 4 по 6
- Кадр 4 принят с ошибкой и проигнорирован *приемником*. Приняв кадр 5 *приемник* фиксирует пропуск кадра 4 и отправляет ***NAK 4*** (Negative Acknowledgment)
- Получив ***NAK 4*** *передатчик* повторно передает кадры с 4 по 6, а затем продолжает сдвигать окно дальше в результате получения ACK 4, 5 и т.д.

Алгоритмы работы узлов Go-back-N ARQ

Приемник, приняв кадр с номером ***n(s)***:

- ✓ Если ***n(s) = n(r)***: посылает ***ACK n(s)***, игнорирует данные (повтор данных)
- ✓ Если ***n(s) = n(r)+1***: посылает ***ACK n(s)***, принимает данные
- ✓ Если ***n(s) > n(r)+1***: посылает ***NAK n(r)+1***, игнорирует данные (пропуск кадра)

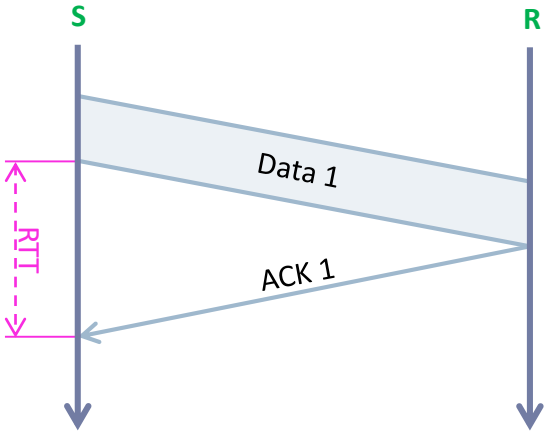
Передатчик:

- ✓ может передавать кадры последовательно в пределах действующего окна, не дожидаясь подтверждения;
- ✓ получив ***ACK x***: сдвигает окно в позицию ***[x+1…x+N]***, продолжает передачу в пределах нового окна с первого не переданного блока;
- ✓ получив ***NAK x***: сдвигает окно в позицию ***[x…x+N-1]***, возобновляет передачу с кадра ***x*** (возвращается к кадру ***x***);
- ✓ при таймауте: повторяет передачу последнего неподтвержденного кадра.

Свойства и недостатки Go-back-N ARQ

- ✓ При достаточном размере окна (и отсутствии ошибок) передача может осуществляться без пауз (достигается 100% утилизация ресурсов канала)
- ✓ В каждый момент времени в состоянии передачи (с момента начала отправки и до получения подтверждения ***ACK***) может находиться максимум ***N*** кадров (***N***-размер окна). Поэтому простой ARQ с остановкой и ожиданием это частный случай Go-back-N с ***N=1***
- ✓ Процедура восстановления ошибочно принятого кадра может инициироваться успешным приемом следующего кадра, что явно быстрее, чем истечение таймаута. Однако для последнего переданного кадра все остается по прежнему (но это не столь критично – ведь все равно пока больше нечего передавать).
- ✓ Примитив ***ACK x*** подтверждает прием всех предыдущих кадров до ***x*** включительно, а ***NAK x*** – до ***x-1*** **!!!** Поэтому потеря не последнего ***ACK*** не сказывается на работе протокола
- ✓ При обнаружении ошибки в единственном кадре повторно передается практически все окно, хотя большая часть могла быть успешно принята.

Выбор оптимальной величины N



- С одной стороны – для эффективного использования канала размер окна ***N*** должен быть достаточно большим
- С другой стороны – величину ***N*** нужно выбирать достаточно малой для того, чтобы минимизировать объем повторной передачи при обработке ***NAK***

- Чтобы канал не простаивал размер окна ***N*** должен позволить передавать кадры в течение всего интервала ***RTT*** (round-trip time)
- Емкость канала: ***Q=RTT·V***
- Минимальный размер окна: ***N=Q/S+1***, примеры:

Среда передачи	Скорость V, байт/сек	RTT, сек	Размер кадра, байт	Емкость Q, байт	Мин. разм. окна N
Ethernet LAN 100 Мбит	12500000	0,0003	1500	3750	4
Ethernet LAN 1 Гбит	125000000	0,0002	1500	25000	18
Канал до Лондона 1 Гбит	125000000	0,04	1500	5000000	3334

Принцип ARQ с выборочным повтором

Для устранения необходимости повторно передавать все кадры окна при ***NAK***, особенно для каналов с большой емкостью (long fat pipes), применяется примитив **запроса выборочного повтора *SREJ x*** (selective reject). В отличие от ***NAK x*** примитив ***SREJ x***:

- не меняет положение окна в *передатчике* и ничего не подтверждает;
- запрашивает повторную передачу только одного кадра ***x***;

Применение ***SREJ*** вместо ***NAK*** существенно усложняет логику *приемника*:

- ✓ *приемнику* необходимо иметь буферную память на все кадры текущего окна и уметь переупорядочивать последовательность принятых кадров;
- ✓ получив в ответ на ***SREJ*** выпавшие из-за ошибок кадры, *приемник* должен выдать ***ACK*** на всю непрерывную последовательность принятых кадров;
- ✓ посылая ***SREJ*** *приемник* откладывает посылку ***NAK*** и должен отсчитывать таймаут и счетчик попыток по не вполне четким критериям;
- ✓ если попытка получить кадры по ***SREJ*** не удалась приемник должен понять это и выдать ***NAK***;
- ✓ слишком длительное ожидание *приемником* ответа на ***SREJ*** может вызвать таймаут в передатчике.

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - 6.1. Функция кадрирования
    - 6.2. Методы контроля правильности передачи данных
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - **6.4. Управление звеном передачи данных**
    - 6.5. Семейство протоколов HDLC
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

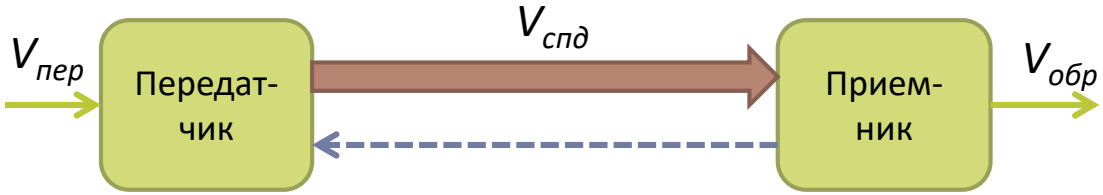
Управление звеном передачи данных

Link Control – набор специфичных управляющих функций, реализуемых в определенных конфигурациях звеньев передачи данных:

- **управление потоком передачи (flow control)**
  Предоставляет возможность приемнику регулировать темп отправки данных передатчиком (чтобы не допустить у себя переполнения буферов)
- **мультиплексирование протоколов**
  Позволяет использовать общее звено передачи для функционирования нескольких сетевых протоколов (IP, ARP,….)
- **управление установкой соединения (DialUp)**
  Необходимо в тех случаях, когда звено передачи данных организуется поверх сети с коммутацией каналов (например, поверх обычной телефонной сети)
- **управление доступом к среде передачи (Media Access Control)**
  Необходимо в тех случаях, когда одна среда передачи сигнала должна совместно использоваться (разделяться) для нескольких потоков передачи данных

Управление потоком передачи данных

При наличии большого объема для передачи, скорость $V_{пер}$, с которой передатчик отправляет данные, ограничена только ресурсами самого канала ($V_{спд}$)



Однако, скорость обработки в приемнике ($V_{обр}$), может быть ограничена. Если $V_{пер} > V_{обр}$, то возникает опасность переполнения приемных буферов. Для исключения такого сценария приемник может использовать один из методов управления потоком (flow control) через обратный канал:

- ✓**Start/Stop (готов/не готов к приему)** – при получении **Stop** передатчик полностью приостанавливает передачу, а по получению **Start** возобновляет. Таким образом передатчик по команде приемника вставляет паузы.
- ✓**Credit Based (основанный на лимите)** – приемник по обратному каналу выдает и периодически обновляет **лимит (credit)** передаваемых данных. В процессе передачи передатчик уменьшает (декрементирует) лимит на объем переданных данных. По исчерпанию лимита передатчик останавливается.

Схема Start/Stop

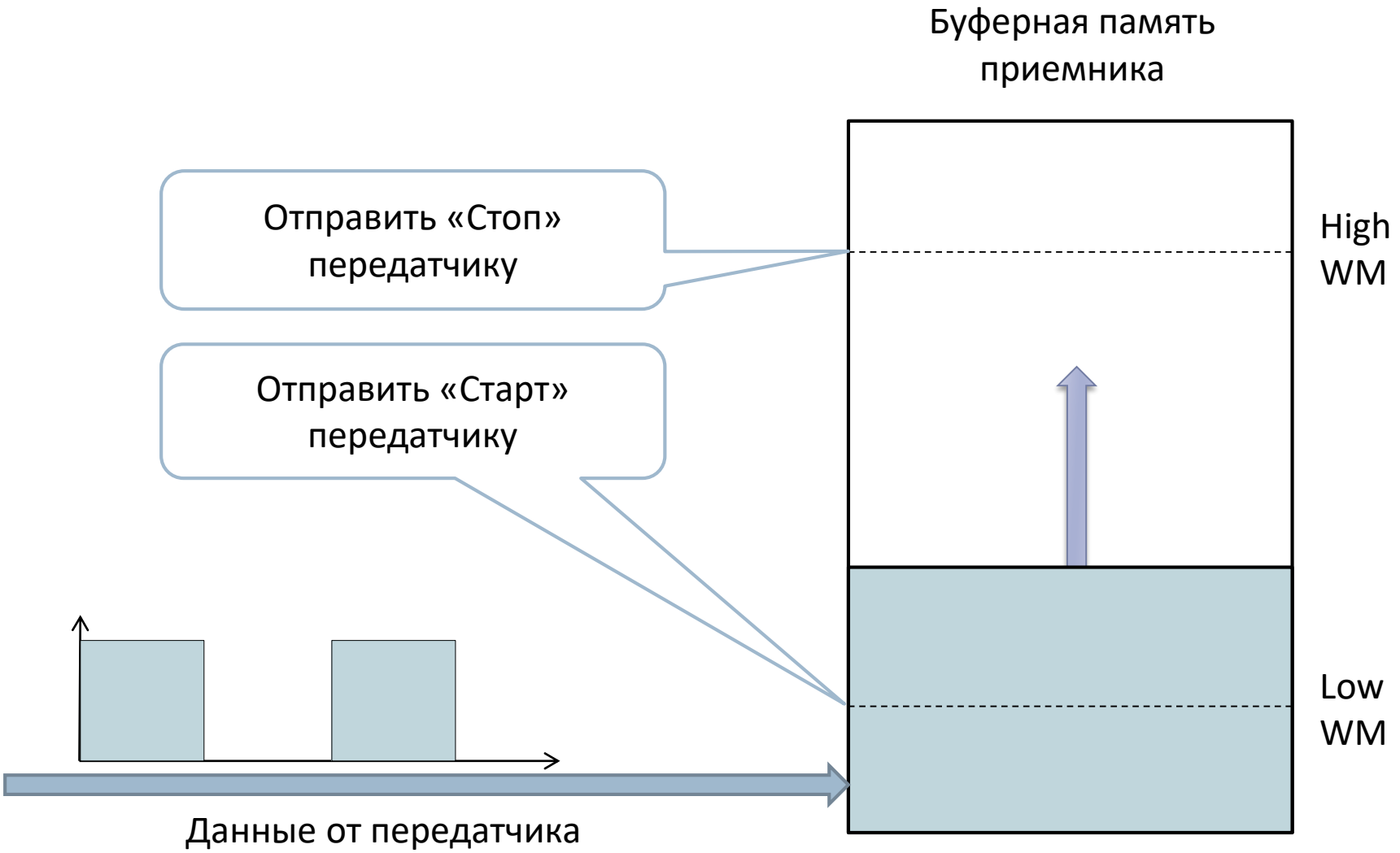
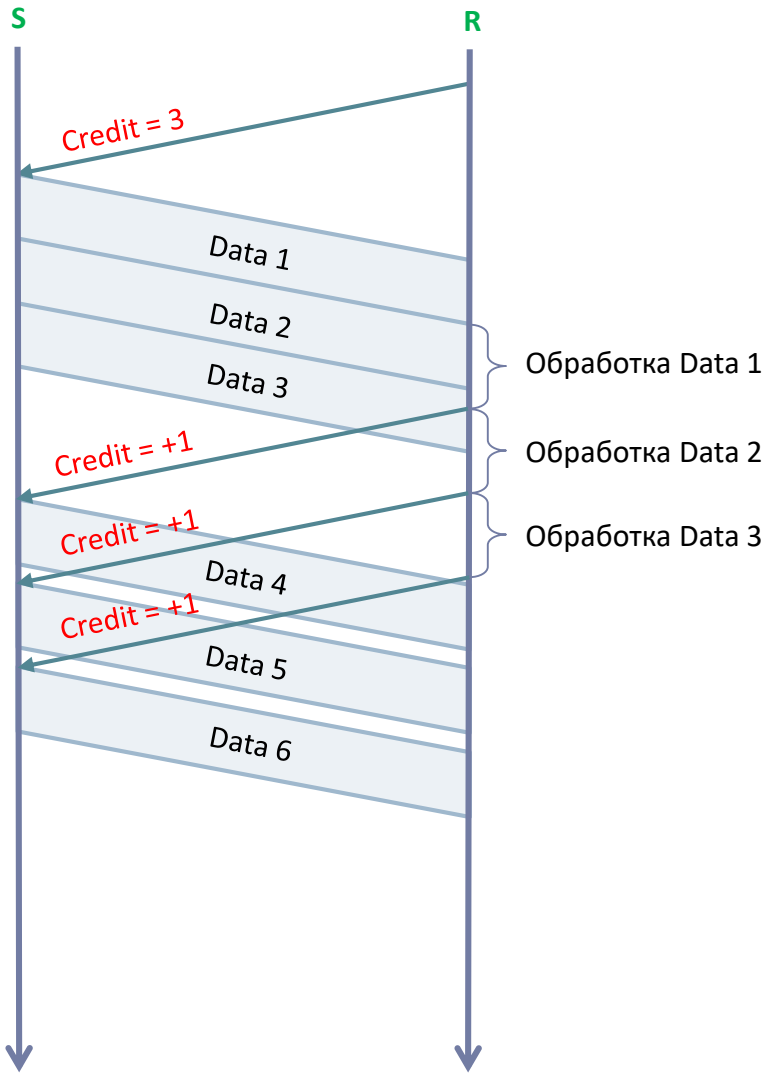


Схема Credit Based

- ✓ Передатчик имеет право отправлять пакеты приемнику только в пределах установленного ему кредита
- ✓ Начальная величина кредита может согласовываться в процессе установки соединения
- ✓ При отправке пакета передатчик уменьшает выделенный ему кредит
- ✓ Если кредит исчерпан (стал равен 0), то передача приостанавливается
- ✓ Приемник пополняет кредит в момент, когда принятый пакет обработан



Управление установкой соединения

Звенья передачи данных могут строиться с использованием:

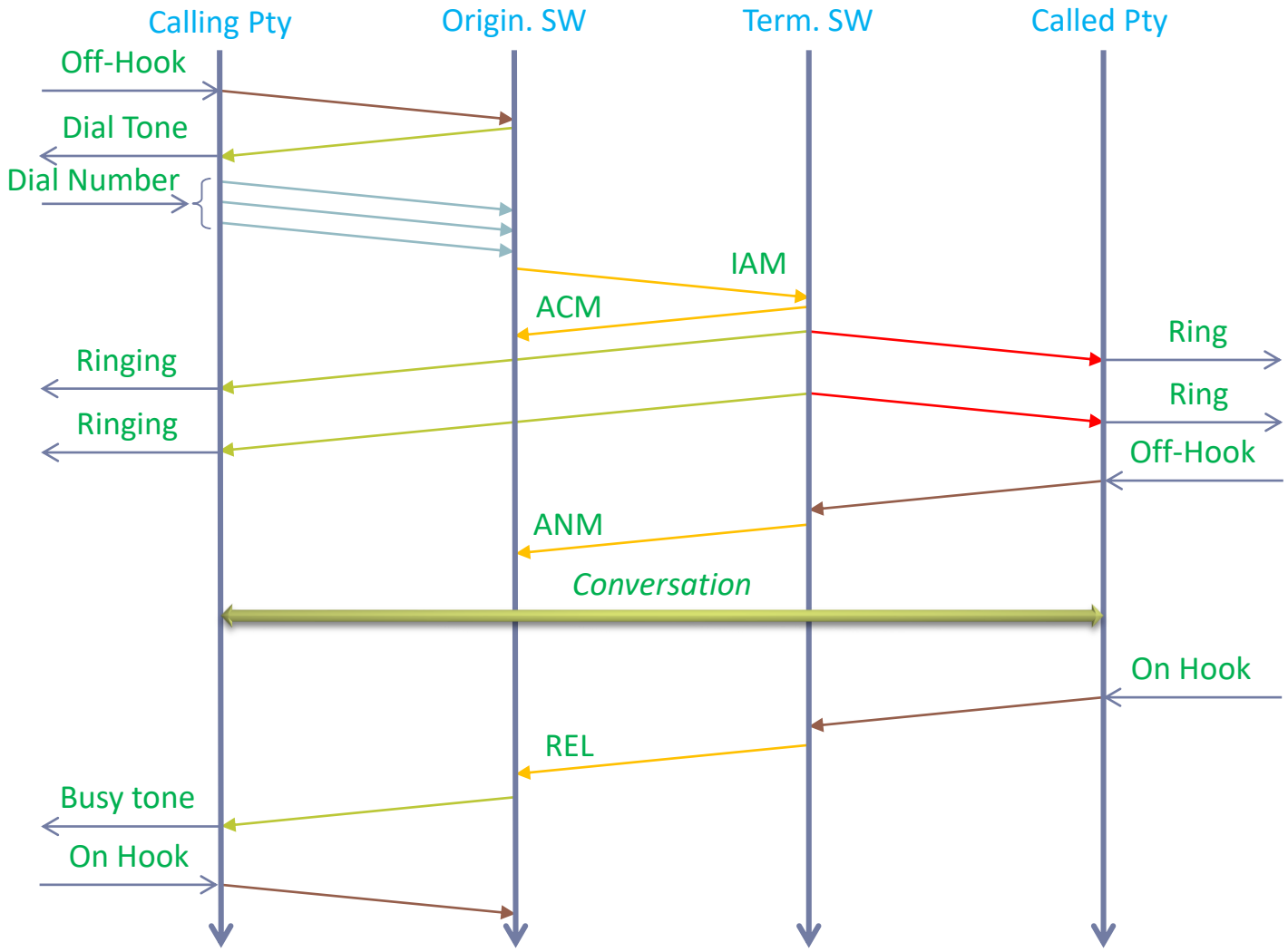
- Постоянных линий (permanent lines);
- Коммутируемых линий (DialUp lines);

Узлы звена, построенного на постоянных линиях, могут начинать обмен данными сразу же после включения: линия соединяет их постоянно.

Перед тем как начать обмен данными через коммутируемую линию узлы сначала должны выполнить процедуру (протокол) установки соединения, актерами которого являются:

- Вызывающий абонент (calling party)
- Вызываемый абонент (called party)
- Исходящая АТС (originating switch)
- Оконечная АТС (terminating switch)

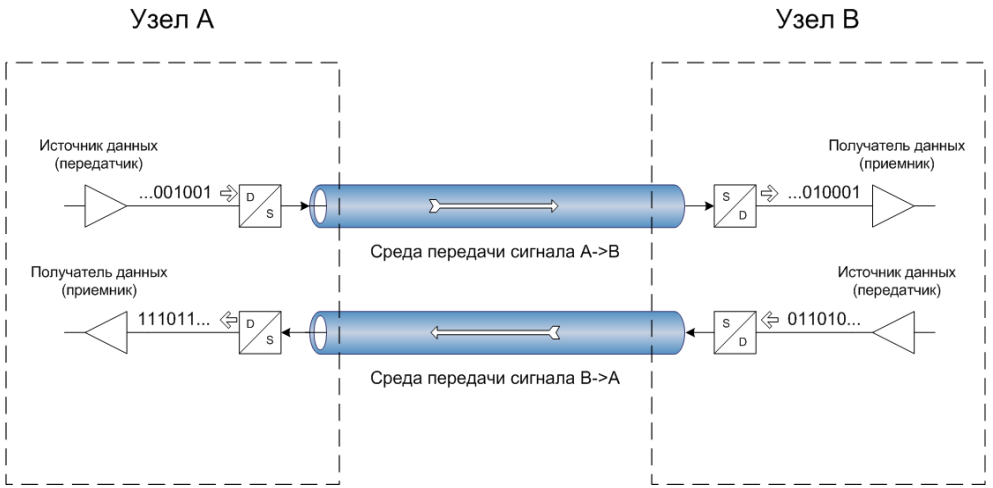
Процедура установки DialUp соединения



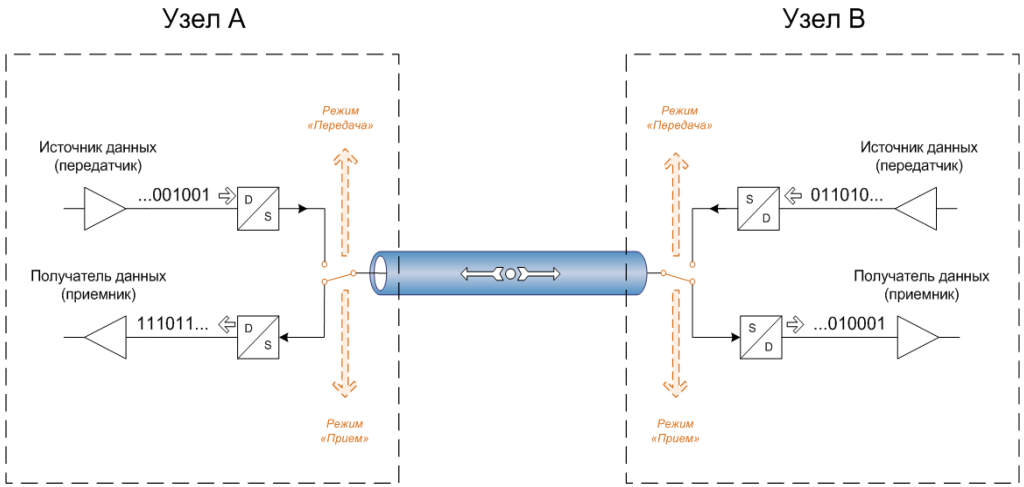
Управление доступом к среде передачи

Media Access Control – процедура доступа нескольких станций к общей разделяемой среде передачи

Дуплексный режим – две отдельных среды передачи.
MAC – не требуется.



Полудуплексный режим – одна среда передачи, способная передавать данные в обоих направлениях, но по очереди. Необходима дисциплина доступа к среде.



Общие свойства процедур (дисциплин) MAC

Полудуплексные звенья могут быть многоточечными, а дуплексные возможны только точка-точка.

Коллизии (collision) возникают в тех случаях, когда две или более станции передают данные одновременно. В результате коллизии передаваемые данные полностью искажаются и не могут быть приняты ни одной из станций.

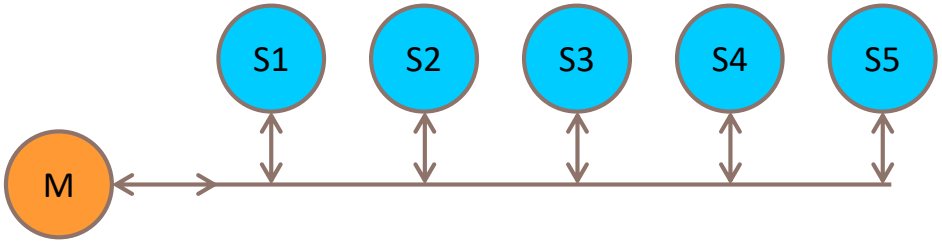
Дисциплины MAC определяют такие способы поведения станций, при которых:

1. Коллизии полностью исключаются:
   - Режим **Master-Slave**
   - Режим **передачи эстафеты** (**Token Bus** и **Token Ring**)
2. Реализуются эффективные способы борьбы с коллизиями и их последствиями:
   - Режим **соперничества**

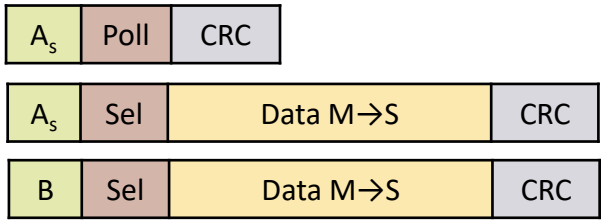
Режим Master-Slave

На звене присутствует одна главная станция (**Master**) и любое количество подчиненных станций (**Slaves**)

Каждой подчиненной станции ***X*** присвоен уникальный адрес ***(A_x).***



- ✓ **Master** имеет возможность передавать по звену кадры трех типов:
  - ➢ Опрос (Poll):
  - ➢ Выборка (Select):
  - ➢ Оповещение (Broadcast):



- ✓ После Poll или Select **Master** переходит в режим приема и ожидает ответа
- ✓ **Slaves** постоянно находятся в режиме приема и получают все кадры **Master**
- ✓ **Slave X**, опознав ***($A_s = A_x$)***, передает в ответ:
  - ➢ На Poll: кадр данных в направлении **Master** (если есть) или **EOT**
  - ➢ На Select: **ACK** или **NAK** в подтверждение приема блока данных от **Master**

Характеристики режима Master-Slave

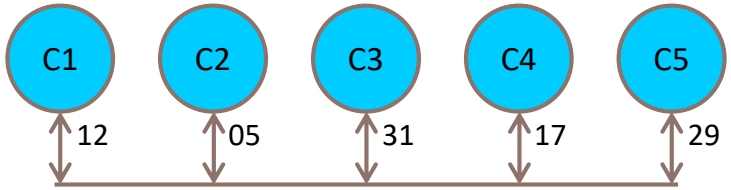
- На звене возможны только три варианта обмена данными:
  - От **Master**-а к конкретному **Slave**
  - От конкретного **Slave** к **Master**-у
  - От **Master**-а одновременно всем активным **Slave**

  Прямой обмен данными между **Slave** **не возможен**
- **Slave** не имеет права начинать передачу по своей инициативе. Любая активность на звене инициируется только **Master**-ом
- Для того, чтобы передать данные **Slave** должен дождаться от **Master**-а прихода **Poll**, адресованного данному **Slave**
- Для реализации двухсторонней передачи **Master** должен постоянно выполнять последовательный циклический опрос всех **Slave** (выполнять **Polling**)
- Отсутствие на звене адресуемого **Slave** или искажение кадров в процессе передачи вызывают тайм-аут приема у **Master**-а и замедляют процедуру **Polling**-а
- При выходе из строя **Master**-а звено целиком становится неработоспособным

Режим маркерной шины (Token Bus)

Все станции на звене являются **равноправными** и в разные моменты времени выступают, как в роли **Master**, так и в роли **Slave**

Каждой станции **X** на звене назначен **уникальный адрес** ***(A_x)***.



- ✓ В каждый момент времени только одна станция обладает правом начать передачу. Это право передается специальным кадром – **маркером (token)**.
- ✓ Получив маркер станция имеет право передать очередной кадр данных (если таковой имеется), а затем обязана передать маркер следующей станции (как эстафетную палочку).
- ✓ В работающей сети маркер передается по логическому кольцу из станций (в примере **31 → 29 → 17 → 12 → 05 → 31 → …..**). Каждая станция в кольце знает ***NID*** (next ID) – адрес следующей станции в логическом кольце
- ✓ Выстраивание логического кольца производится в **процедуре автоматической реконфигурации**. Она построена на таймаутах, величина которых (обратно) пропорциональна адресу станции.

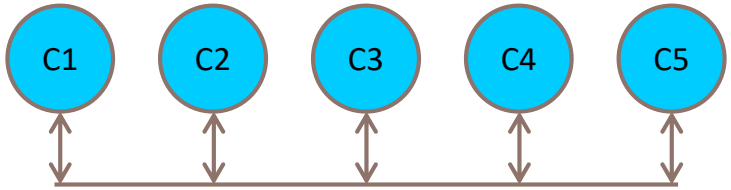
Характеристики режима Token-Bus

- Реконфигурация кольца запускается по тайм-ауту в результате потери маркера, которая может возникать:
  - При отключении станции, входившей в кольцо
  - При искажении маркера из-за ошибки передачи
  - При подключении новой станции к шине

  В последнем случае вновь подключившаяся станция искусственно вызывает потерю маркера путем организации **управляемой коллизии**.
- Все станции, которые в данный момент не владеют маркером, ведут себя как **Slave**, т.е. прослушивают шину, принимают адресованные к ним кадры и отправляют подтверждения (если требуется)
- Благодаря логическому кольцу каждая станция получает равный шанс отправить (передать) накопившиеся данные.
- Несмотря на то, что коллизии практически сведены к нулю (событие подключения новой станции к кольцу происходит редко), степень использования среды передачи остается невысокой.

Режим соперничества

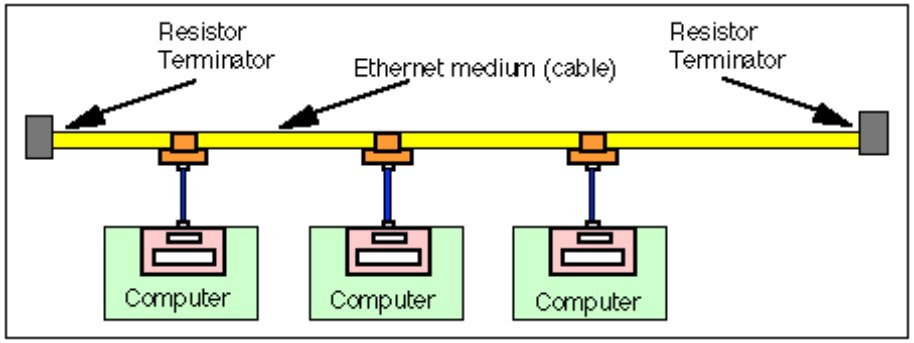
Все станции на звене являются **равноправными** и сочетают в себе роли как **Slave**, так и **Master**. Причем, каждая станция самостоятельно принимает решение о том, когда начинать передачу (стать Master устройством).



- ✓ При работе в таком режиме **коллизии** не исключаются и **являются частью нормального процесса функционирования** звена.
- ✓ При этом используются меры по снижению негативного эффекта от коллизий:
  - ➢ Минимизируется вероятность возникновения коллизий
  - ➢ Сокращается время действия коллизии
  - ➢ Принимаются меры по автоматическому восстановлению данных, потерянных в результате коллизии
- ✓ На сегодняшний момент наиболее распространенным и технически совершенным вариантом сетевой технологии на базе режима соперничества является **Ethernet**

Технология Ethernet

- ✓ Изобретена в лаборатории XeroxPARC в 1973-1974 году (Robert “Bob” Metcalfe, впоследствии организовал компанию 3COM – первого производителя оборудования Ethernet)
- ✓ Изначально использовала «толстый» коаксиальный кабель
- ✓ Конструкция простой широковещательной шины: все станции подключены параллельно к одному кабелю - сетевому сегменту



Метод доступа к среде, применяемый в Ethernet

Технология Ethernet работает в режиме соперничества и использует дисциплину (метод) доступа к среде:

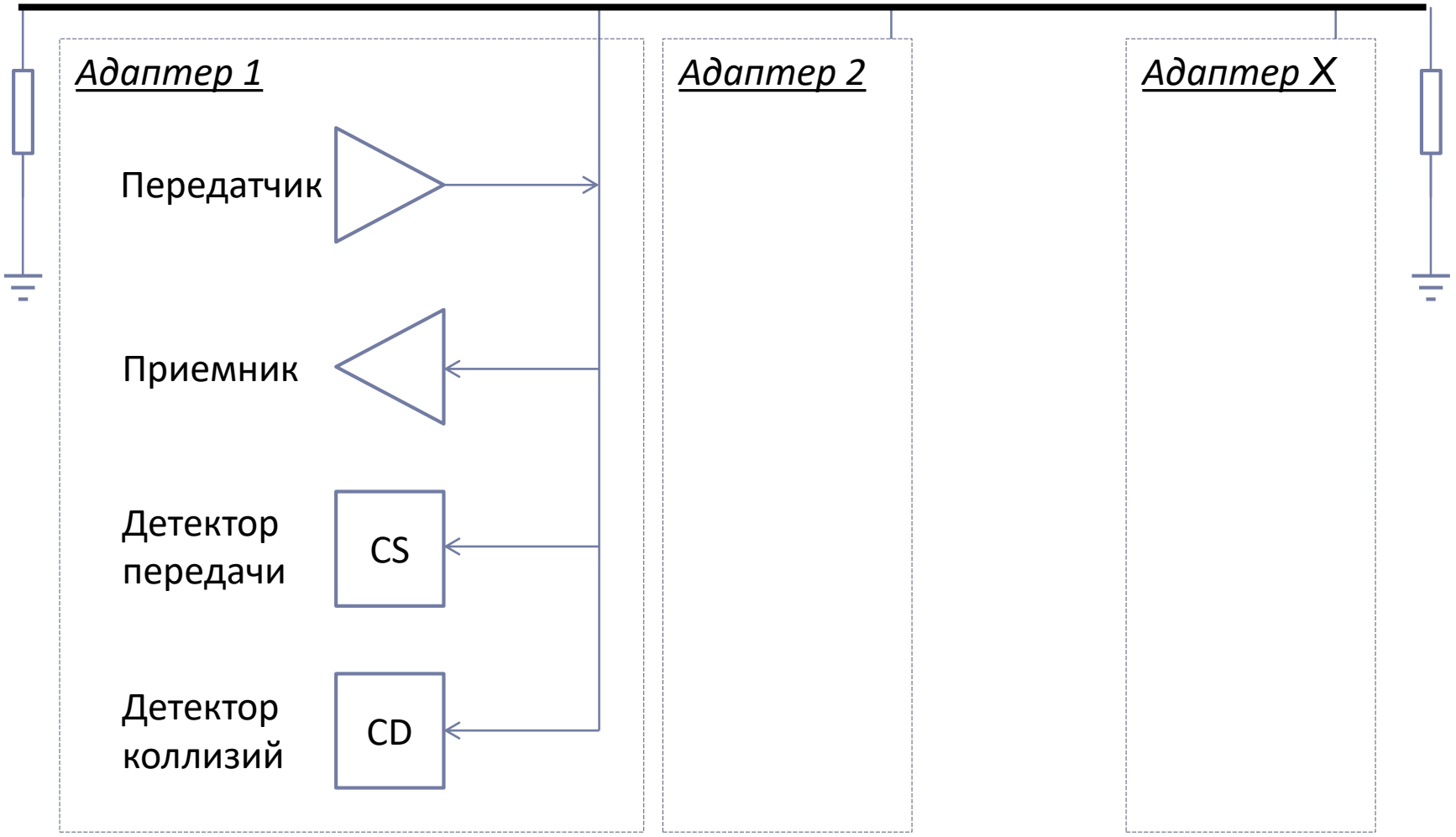
- **CSMA/CD** (**C**arrier-**S**ense **M**ultiple **A**ccess / **C**ollision **D**etection)
- **МДПН/ОК** (**М**ножественный **Д**оступ с **П**рослушиванием **Н**есущей / **О**бнаружением **К**оллизий)

Для реализации этой дисциплины каждая станция Ethernet оборудуется дополнительными устройствами, подключенными непосредственно к шине:

- ✓ Детектор несущей – электрическая схема, фиксирующая наличие активной передачи данных по шине
- ✓ Детектор коллизии – электрическая схема сигнализирующая о том, что одновременно ведут передачу две или более станции

Эти дополнительные устройства не обрабатывают данные – они просто измеряют уровень электрического напряжения на шине и работают даже когда станция находится в режиме передачи.

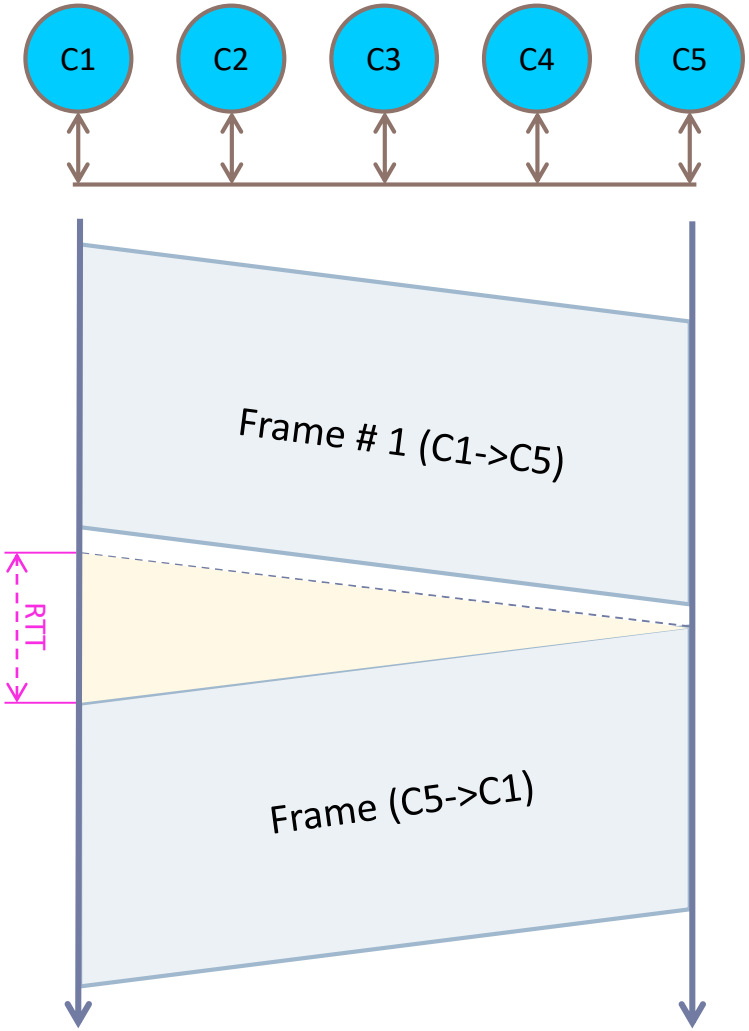
Логическая схема адаптера Ethernet



Алгоритм доступа к среде Ethernet

1. Перед началом передачи убедиться, что детектор несущей не активен. В противном случае ждать сначала освобождения среды, а затем истечения минимального межкадрового промежутка (Inter-Frame Gap).
2. Начать передачу. В процессе передачи кадра контролировать сигнал с детектора коллизий.
3. При срабатывании детектора коллизии:
   - если еще не достигнут минимальный размер кадра, то передавать псевдо шумовой сигнал, чтобы все станции зафиксировали коллизию
   - Выключить передачу. Увеличить счетчик попыток ***C***. Если уже предпринято 16 попыток, то зафиксировать невозможность передачи кадра и закончить.
   - Выдержать паузу величина которой выбирается псевдо случайным образом как $slot\_time \cdot rand(0..2^c-1)$. Таким образом максимально возможная величина задержки при каждом повторе увеличивается экспоненциально вплоть до 1023
   - Повторить с пункта 1

Механизм возникновения коллизии



Характеристики метода CSMA/CD

- Благодаря детектору несущей коллизия может возникнуть только в начале кадра в течение round trip time (RTT). В остальное время коллизии исключены
- Благодаря детектору коллизий передача (и коллизия) заканчивается так быстро, как это возможно и выполняется автоматическое повторение передачи кадра (retransmission)
- Вероятность получения повторной коллизии сведена к минимуму за счет применения генераторов случайных чисел
- В целом метод позволяет достичь 90-100% утилизации в оптимальных условиях, однако эффективность существенно снижается:
  - ✓ При увеличении максимального диаметра сети (расстояния между двумя наиболее отдаленными друг от друга станциями);
  - ✓ При росте количества станций;
  - ✓ При росте объема трафика на передачу

План курса

1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - 6.1. Функция кадрирования
    - 6.2. Методы контроля правильности передачи данных
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - 6.4. Управление звеном передачи данных
    - **6.5. Семейство протоколов HDLC**
    - 6.6. Семейство протоколов PPP
7. Протоколы транспортного уровня
8. Технологии WWW

Протокол HDLC

HDLC (High-level Data Link Control) – протокол *высокоуровневого* управления звеном передачи данных. Является (идейным) прародителем большинства современных протоколов канального уровня (ITU LAPx, IEEE 802.1 LLC, Cisco HDLC, PPP и др.)

Использует схемы кадрирования с флагами (***0x7E***), как в синхронном, так и в асинхронном режимах

Общая структура **кадра HDLC**:

Address (адрес узла): фиксированная длина 1 байт и более, на конкретных звеньях может отсутствовать

Control (управляющее поле): 1 или 2 байта (в зависимости от режима) определяет тип кадра, а также служебную информацию протокола

Information (передаваемые данные): переменная длина, поле присутствует не во всех типах кадров

Виды кадров и структура поля Control

I-frame - (информационный кадр)

S-frame - (управляющий кадр)

U-frame - (ненумерованный кадр)

b7	b6	b5	b4	b3	B2	B1	B0
N(r)			P/F	N(r)			**0**
N(r)			P/F	Stype		**0**	**1**
Utype[2:4]			P/F	Utype[0:1]		**1**	**1**

Поле N(r) – номер кадра, который станция-отправитель ожидает принять

Поле N(s) – номер передаваемого кадра

Бит P/F (Poll/Finish) – сигнализирует о том, что станция-отправитель предлагает станции-адресату продолжить диалог (условная фраза «Прием»)

Поле Stype – указывает вид управляющего (супервизорного) кадра:

(***00***) – **RR** (Receive Ready): «принято, продолжайте с *N(r)*»
(***01***) – **RNR** (Receive Not Ready): «принято до *N(r)-1*, подождите»
(***10***) – **REJ** (Reject): «повторите начиная с *N(r)*»
(***11***) – **SREJ** (Selective Reject): «повторите кадр *N(r)*»

Поле Utype – указывает вид ненумерованного кадра

Режимы обмена данными

Дейтаграмный режим – передача без ARQ:

Обмен кадрами **UI** (unnumbered information, Control=0x03)

Режим обмена нумерованными кадрами – передача с ARQ:

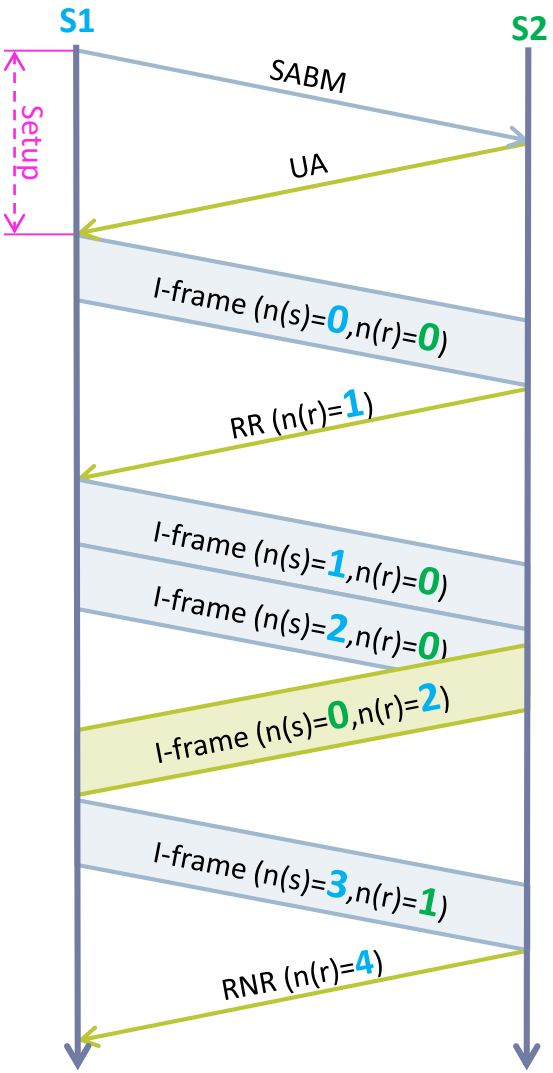
Данные передаются в пронумерованных I-кадрах под управлением ARQ протокола. В качестве обратной связи (ARQ-примитивов) могут использоваться как S-кадры, так и I-кадры противоположного направления. Перед началом передачи данных выполняется процедура установки соединения.

Предусмотрено три вида управления:

- **NRM** (normal response mode) – Master/Slave
- **ABM** (asynchronous balanced mode) – дуплексный двухточечный режим
- **ARM** (asynchronous response mode) – режим соперничества

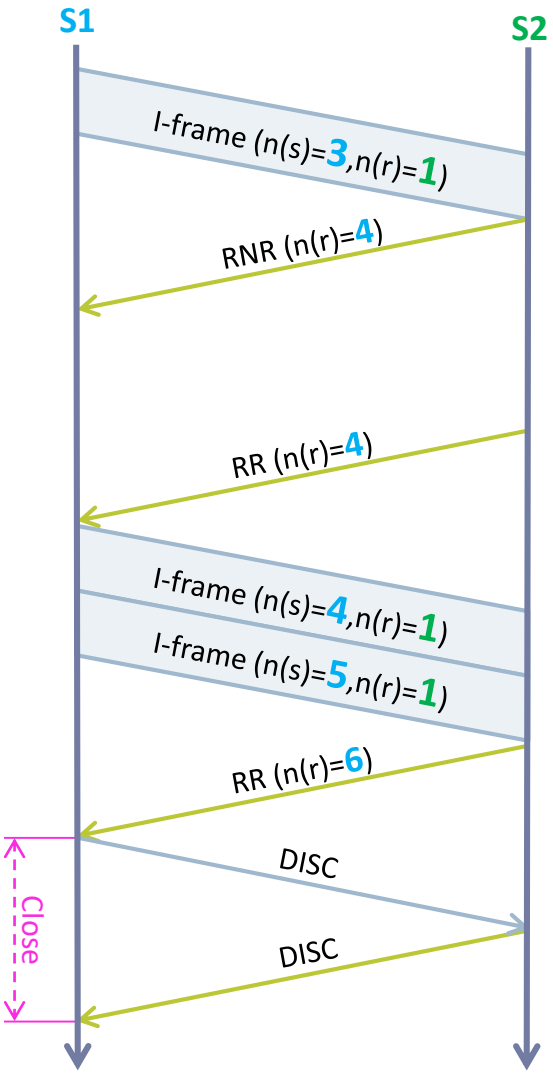
* Режимы NRM и ARM уже практически не встречаются.

Сценарий обмена в режиме ABM



- Перед началом обмена данными станции устанавливают соединение обменом **U-frame**:
  - **SABM(E)** : Set ABM mode (extended)
  - **UA** : unnumbered acknowledgement
- Затем станции могут начинать передавать друг другу кадры данных (**I-frames**), причем каждая станция присваивает последовательный номер **N(s)** каждому отправляемому кадру
- Подтверждение приема (ACK) может высылаться в виде **S-frame RR** или **RNR**. **N(r)** показывает номер кадра, ожидаемого станцией-отправителем
- При наличии данных встречного направления подтверждение приема может доставляться в **I-frames**

Управление потоком и закрытие соединения



- В процессе обмена станция может запросить приостановку передачи данных (паузу) путем отправки **S-frame RNR**.
- Возобновление передачи происходит после получения **S-frame RR**, как правило, с тем же самым номером **N(r)**.
- Завершение соединения выполняется путем отправки **U-frame DISC**.

План курса

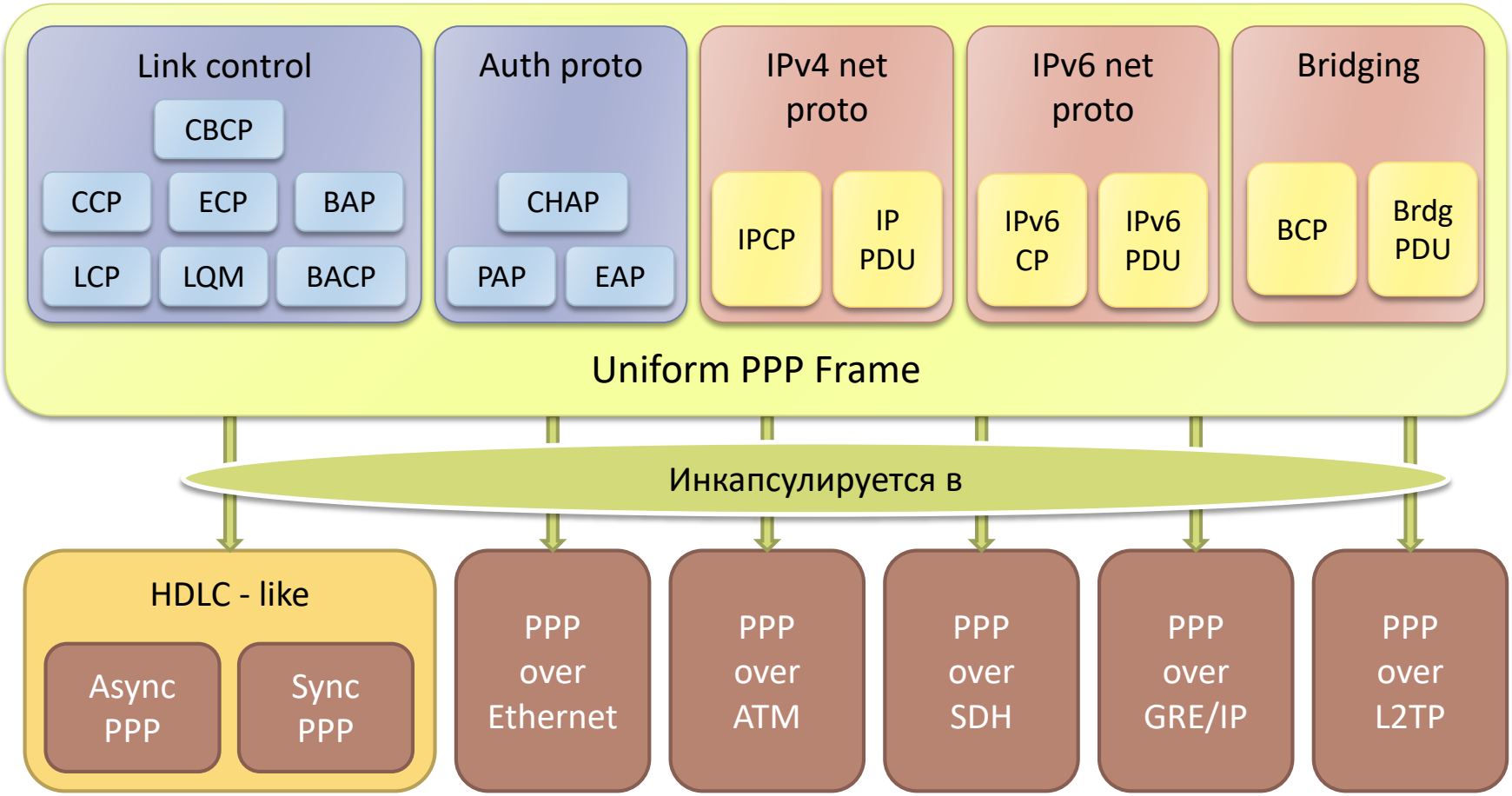
1. Введение в компьютерные сети
2. Основные методы построения СПД
3. Архитектура Internet Protocol Suite (TCP/IP)
4. Архитектура модулей физического уровня
5. Технологии беспроводных сетей
6. Архитектура модулей канального уровня
    - 6.1. Функция кадрирования
    - 6.2. Методы контроля правильности передачи данных
    - 6.3. Восстановление данных, искаженных в процессе передачи
    - 6.4. Управление звеном передачи данных
    - 6.5. Семейство протоколов HDLC
    - **6.6. Семейство протоколов PPP**
7. Протоколы транспортного уровня
8. Технологии WWW

Протокол PPP

Point to Point Protocol – это целое семейство протоколов, позволяющих организовывать разнообразные варианты двухточечных звеньев передачи данных (звеньев «точка – точка»):

- ✓ звенья PPP могут организовываться как по физическим последовательным каналам связи (Synchronous and Asynchronous Serial Links), так и поверх других протоколов канального уровня:
  - PPPoE – PPP over Ethernet
  - PPPoA – PPP over ATM
  - PPPoS – PPP over SONET/SDH
- ✓ кадры PPP могут передаваться (туннелироваться) поверх сетевых или транспортных протоколов (GRE/PPTP, PPP over L2TP, PPP over SSH и т.п.)
- ✓ PPP поддерживает передачу пакетов различных сетевых протоколов(IP, ARP, IPX, NetBIOS, AppleTalk…), а также других видов трафика (Ethernet Frames, Serial Data, TRILL…)
- ✓ PPP имеет встроенную поддержку для многих полезных features: авто конфигурации, аутентификации, контроля качества канала, multi-link, ARQ, компрессии данных, шифрования и др.

Архитектура семейства PPP

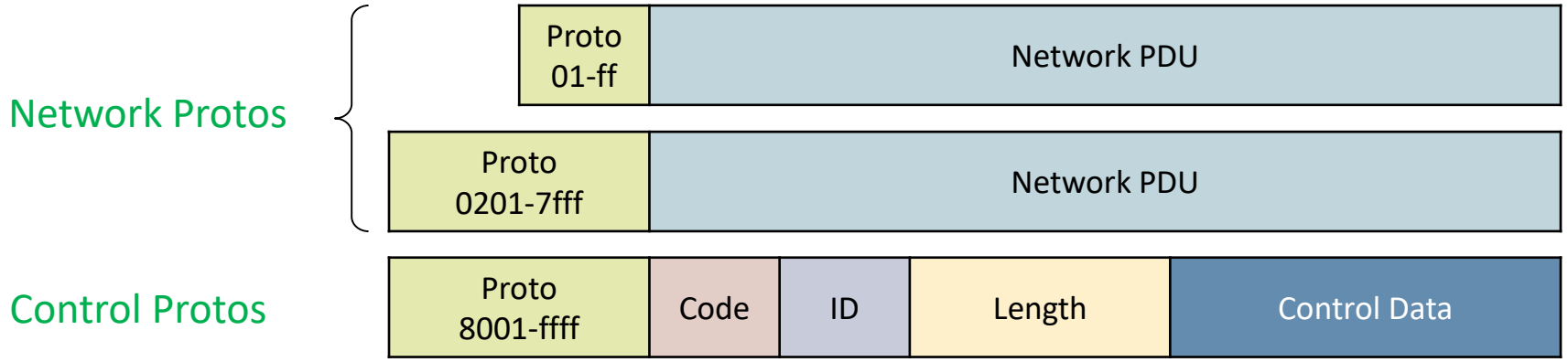


Структура кадров PPP

В рамках PPP определено множество протоколов из двух больших групп:

- протоколы передачи данных (Network Protocols)
- служебные протоколы управления (Control Protocols – **CP**), подразделяемые:
  - на протоколы управления звеном (Link Control Protocols – **LCP**)
  - и на протоколы управления сетевыми протоколами (Network Control Protocols – **NCP**)

Каждому протоколу присваивается уникальный одно- или двухбайтовый код.



Устройство управляющих протоколов (xCP)

Большинство управляющих протоколов построены по одной схеме – взаимодействующие узлы посылают друг другу управляющие пакеты (**Control Data**), одного из следующих типов:

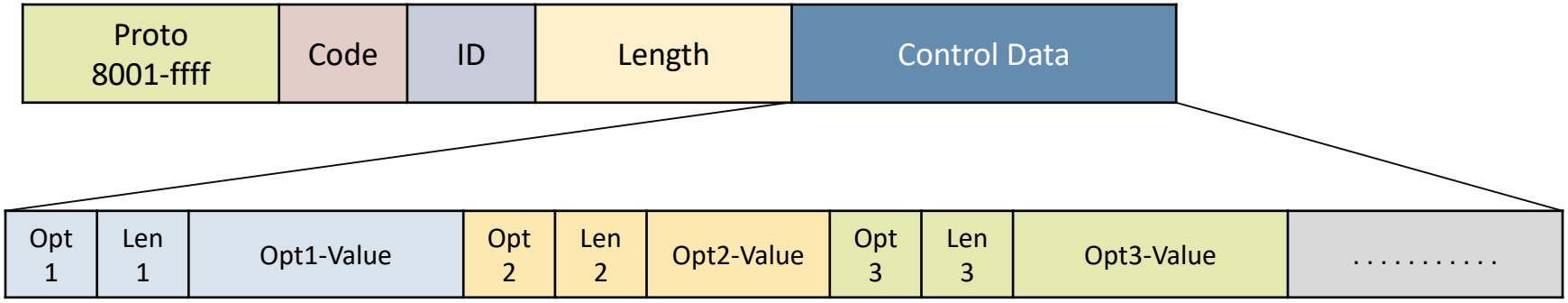
Code	Packet Type	Описание	Функция
1	Configure-Request	Запрос на согласование	Согласование параметров (option negotiation)
2	Configure-Ack	Подтверждение	
3	Configure-Nak	Уточнение	
4	Configure-Reject	Отклонение	
5	Terminate-Request	Запрос отключения	Завершение соединения
6	Terminate-Ack	Подтверждение отключения	
7	Code-Reject	Неизвестный код пакета	Механизм расширения
8	Protocol-Reject	Неизвестный протокол	
9	Echo-Request	Запрос эха	Диагностика соединения
10	Echo-Reply	Эхо ответ	
11	Discard-Request	Запрос подавления	

Необходимость согласования параметров

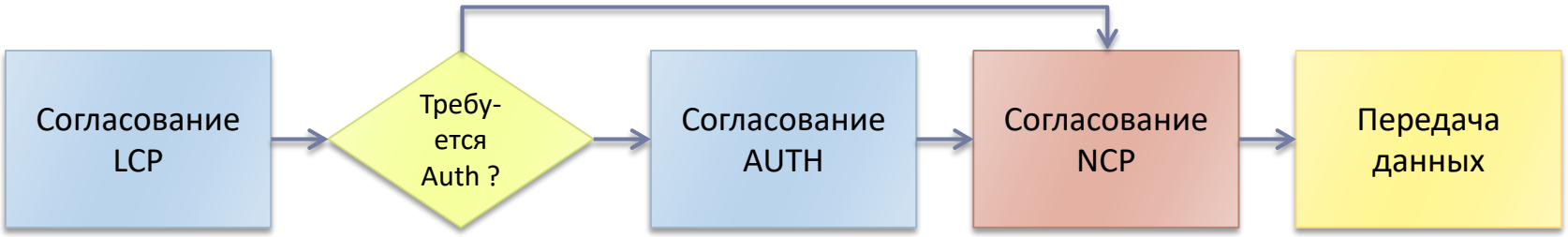
Одной из основных функций протоколов управления является согласование значений параметров (**option negotiation**) у взаимодействующих станций.

С целью достижения гибкости и универсальности PPP способен функционировать в множестве различных режимов, зачастую не совместимых друг с другом. Для того, чтобы взаимодействующие станции смогли нормально функционировать, им необходимо «подстроится» друг под друга и подобрать совместимый режим.

Режим работы станции PPP определяется действующими значениями стандартизованных параметров (**options**), которые могут независимо устанавливаться на каждой станции.



Процедура установки соединения PPP



Пример диалога согласования при установке соединения с Интернет-провайдером

(вид со стороны абонента)

1. sent [LCP Req mru=1480 magic=0xbf2b3179]
2. rcvd [LCP Req mru=1480 auth=chap-MD5 magic=0x5a00382]
3. sent [LCP Ack mru=1480 auth=chap-MD5 magic=0x5a00382]
4. rcvd [LCP Ack mru=1480 magic=0xbf2b3179]
5. rcvd [CHAP Challenge (b568a5782b3c9302bb8e2360efeea9a2), name = "Huawei"]
6. sent [CHAP Response (c6192120b4ac3edc49b65c327d4af208), name = “user123"]
7. rcvd [CHAP Success "Authentication success,Welcome!"]
8. sent [IPCP Req addr=0.0.0.0 dns1=0.0.0.0 dns3=0.0.0.0]
9. rcvd [IPCP Req addr=178.234.192.1]
10. sent [IPCP Ack addr=178.234.192.1]
11. rcvd [IPCP Nak addr=178.234.195.129 dns1=195.34.224.1 dns3=195.34.224.2]
12. sent [IPCP Req addr=178.234.195.129 dns1=195.34.224.1 dns3=195.34.224.2]
13. rcvd [IPCP Ack addr=178.234.195.129 dns1=195.34.224.1 dns3=195.34.224.2]

Фаза LCP: Согласование линейного протокола

Фаза AUTH: Аутентификация абонента

Фаза IPCP: Согласование параметров сетевого протокола

Процедура согласования параметров

- ✓ При установке соединения станция **A** инициализируя протокол xCP посылает управляющий пакет: ***Conf-Req: Opt1=V1, Opt2=V2, Opt3=V3….***
- ✓ Станция **B** анализирует полученный список опций и приходит к выводу:
  - Значение ***Opt1*** должно быть ***V1A*** → send ***Conf-NAK: Opt1=V1A***
  - Станция не знает, что такое ***Opt2*** → send ***Conf-Reject: Opt2=V2***
- ✓ Станция **A** рассматривает пришедшие ответы, корректирует список опций и повторно отправляет: ***Conf-Req: Opt1=V1A, Opt3=V3…..***
- ✓ Станция **B** подтверждает список опций: ***Conf-Ack: Opt1=V1A, Opt3=V3…..***

  - ➢ Данная процедура согласования выполняется в обе стороны одновременно: **A** согласует набор своих опций у **B** и наоборот;
  - ➢ Чтобы не запутаться ***Conf-Req*** и все ответы на него должны иметь одинаковое значение **ID**, которое выбирается псевдо-случайным образом
  - ➢ Каждый из протоколов xCP должен успешно выполнить согласование тех параметров, которыми он управляет

Протоколы PPP и их опции (выборочно)

Proto	Функция	Опция	Назначение
LCP (c021)	Установка параметров соединения и режима аутентификации	MRU=nn	Максимальный размер принимаемого пакета
		ACCM=cccccc	Маска модифицируемых символов
		AUTH=p	Необходимость аутентификации
		MAGIC=n	Magic number
		ACFC	Исключение полей Addr и Control из кадра HDLC
		CALLBACK=dddd	Запрос обратного звонка
IPCP (8021)	Установка настроек IP интерфейса	IP-Addr=x.x.x.x	Желаемый IP адрес
		IP-Comp-Proto=p	Алгоритм компрессии заголовков
		Primary-DNS=x.x.x.x	Адрес DNS сервера
		Primary-NBNS=x.x.x.x	Адрес WINS сервера
PAP	Аутентификация	Имя/пароль	Идентификация станции с которой устанавливается соединение. Может быть односторонней и обоюдной.
CHAP		Challenge/Handshake	
EAP		Много опций