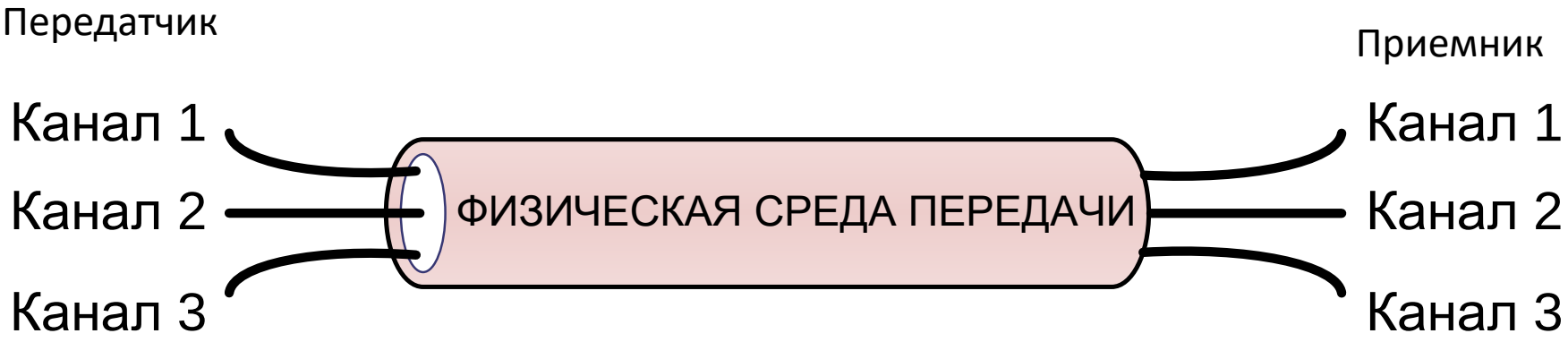
КАНАЛЬНЫЙ УРОВЕНЬ СЕТЕЙ ЭВМ И ТЕЛЕКОММУНИКАЦИЙ



Требуется обеспечить передачу данных от источника(ов) к приемнику(ам), т.е. сформировать между ними **КАНАЛ(Ы) ПЕРЕДАЧИ**.

Режимы работы каналов передачи

- **Однонаправленный (simplex) канал** – передача данных осуществляется в одну сторону (от источника к приемнику);
- **Разделяемый (half-duplex)** – источник и приемник по очереди меняются местами;
- **Двунаправленный (full-duplex)** – данные могут одновременно передаваться от источника к приемнику и от приемника к источнику*.

*) По сути, двунаправленный режим работы реализуется путем организации двух параллельных однонаправленных каналов.

Проблема 1. Синхронизация приемника и передатчика

Действия передатчика и приемника определяются генератором тактовой частоты.

Идеального совпадения частот у двух генераторов НЕ БЫВАЕТ.

Проблема 1. Синхронизация приемника и передатчика (продолжение)

Задача – обеспечить гарантированную передачу последовательности бит заданной длины.

Проблема 1. Синхронизация приемника и передатчика (продолжение)

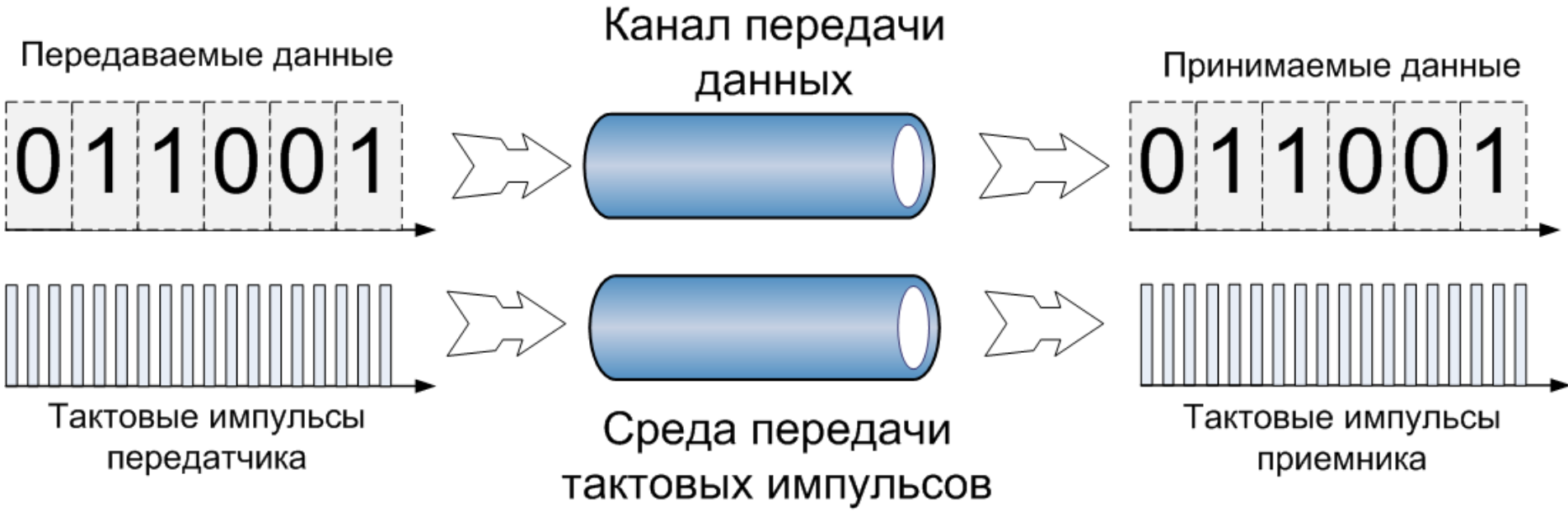
Способ решения 1 – Асинхронная передача.

Проблемы:

- Время передачи данных увеличивается (за счет синхронизации).
- Необходимо обеспечить механизм гарантированного определения синхронизирующих бит.
- Данные обычно передаются по 7 или 8 разрядов (байтами)

Проблема 1. Синхронизация приемника и передатчика (продолжение)

Способ решения 2 – Использовать приемником и передатчиком один генератор тактовых импульсов.

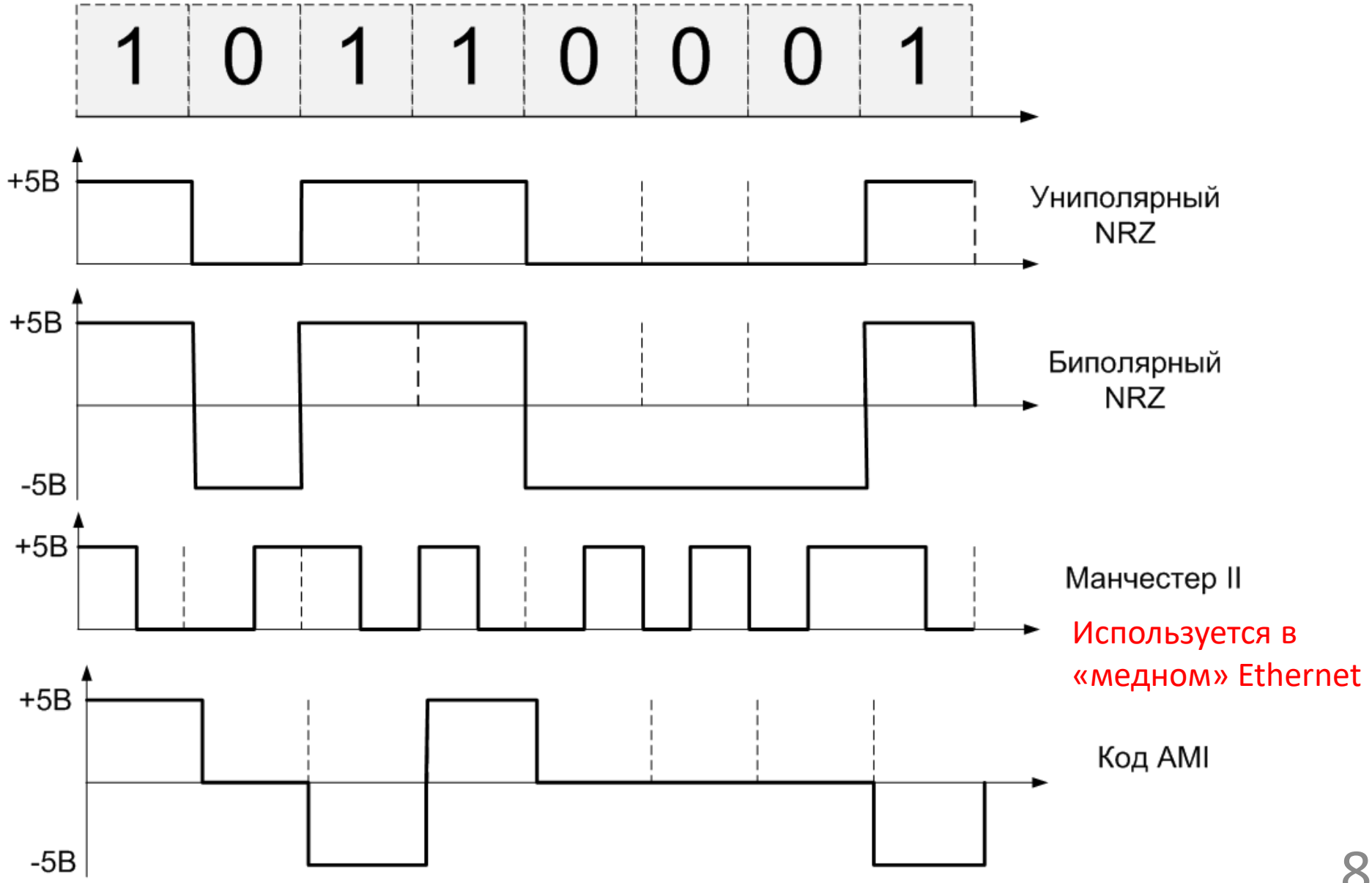


Ограничения связаны с тем, что передача тактовых импульсов должна быть гарантированной.

- Ограничения по длине среды передачи импульсов.

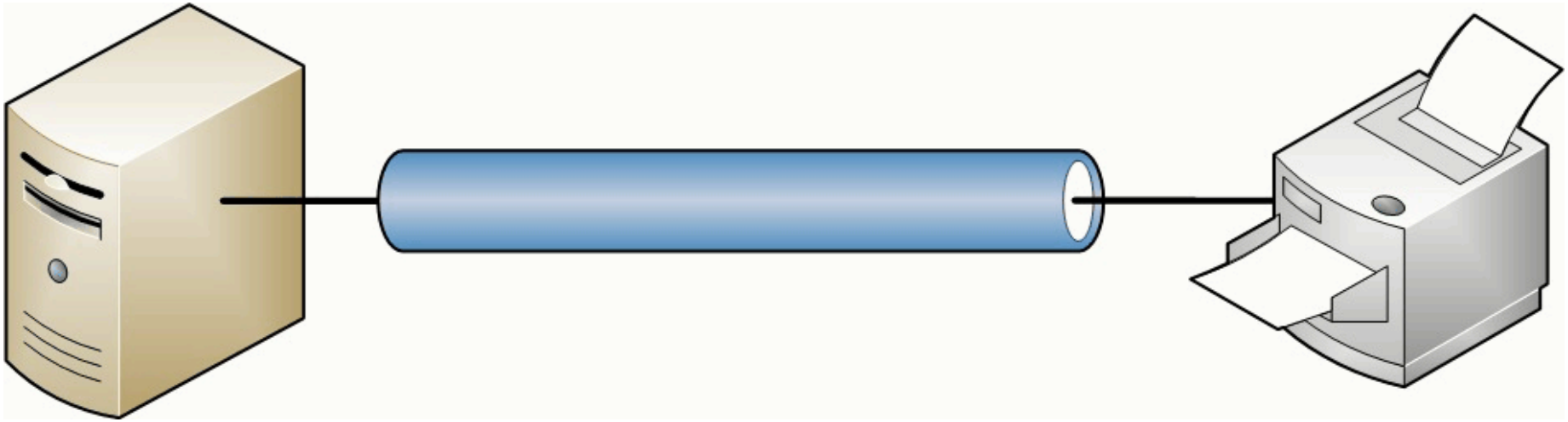
Проблема 1. Синхронизация приемника и передатчика (продолжение)

Способ решения 3 – Интеграция синхронизирующей информации в сигнал.



Проблема 2. Управление потоком информации

Процесс передачи информации предполагает, что приемник готов её принять.



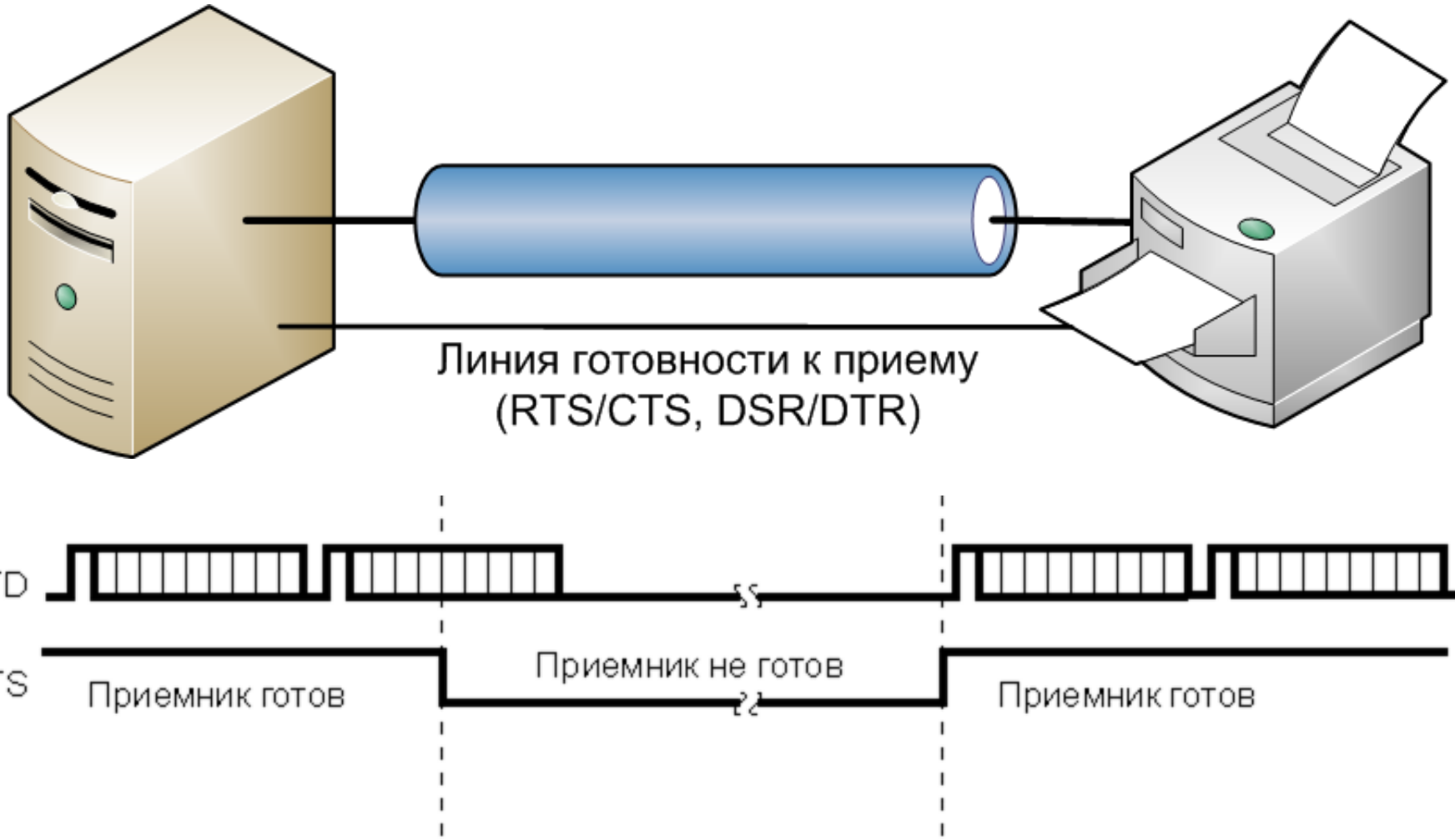
Время на обработку принятого кадра в принимающем устройстве может быть значительно больше времени, требующегося для передачи следующего кадра.

В результате, следующий передающийся кадр будет теряться, т.к. принимающее устройство не будет готово его принять, а передающее устройство ничего об этом не знает.

Управление потоком – возможность принимающей стороны ограничить объем или скорость передачи данных по каналу связи.

Проблема 2. Управление потоком информации

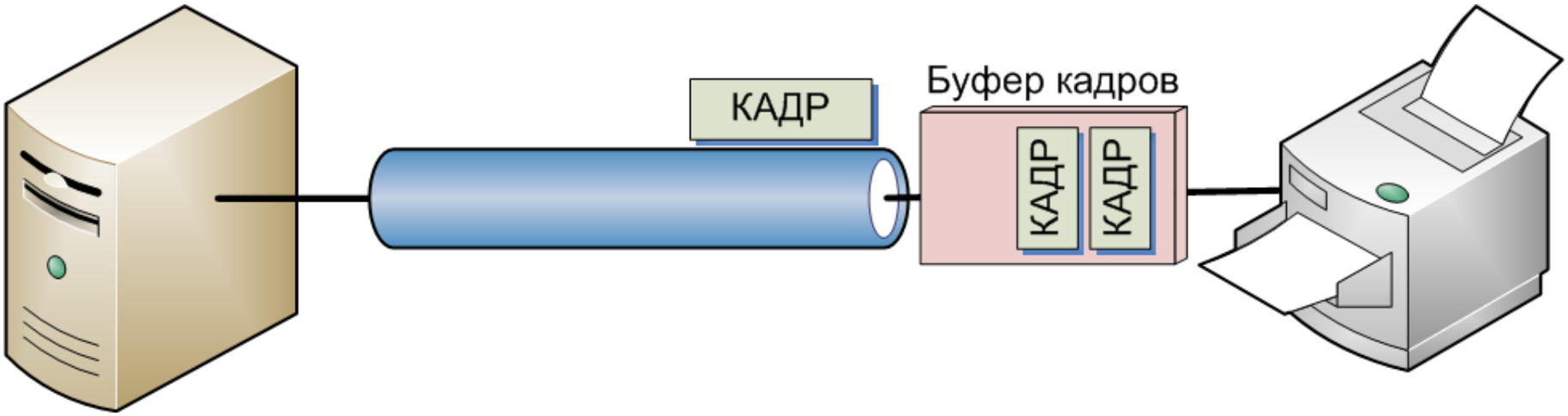
Способ решения проблемы 1 – Аппаратурное управление потоком.



Очередной кадр передается только при наличии сигнала о готовности к приему.

Проблема 2. Управление потоком информации

Способ решения проблемы 2 – Организация в принимающем устройстве буфера для входящих сообщений.



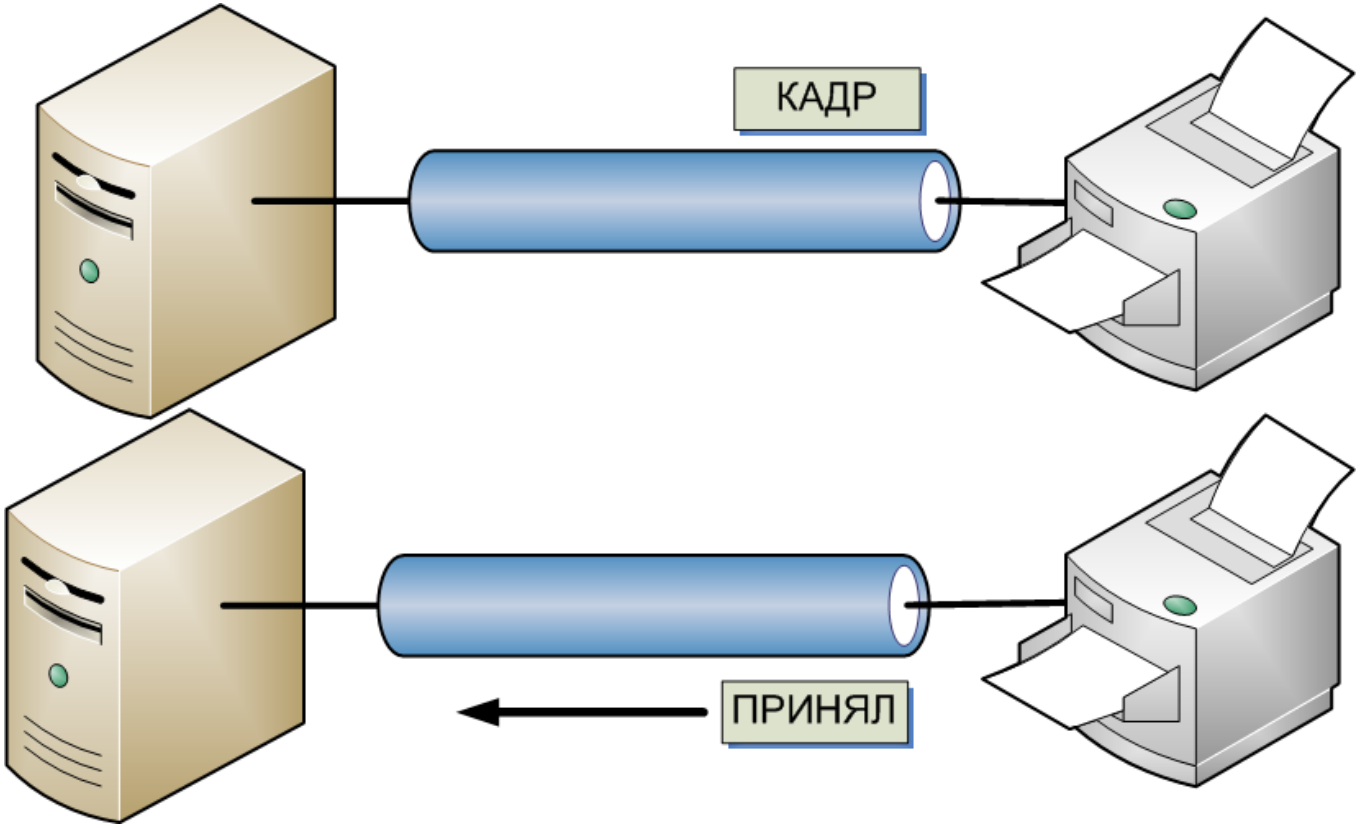
Проблема решена частично: необходимо обеспечить такую длину буфера, чтобы обеспечить гарантированный прием данных при любой продолжительности обработки кадра.

Из-за технических ограничений буфер бесконечной длины организовать нельзя ☺.

Проблема 2. Управление потоком информации

Способ решения проблемы 3 – Режим «Остановка» – «Ожидание».

Шаг 1. Передается кадр с данными

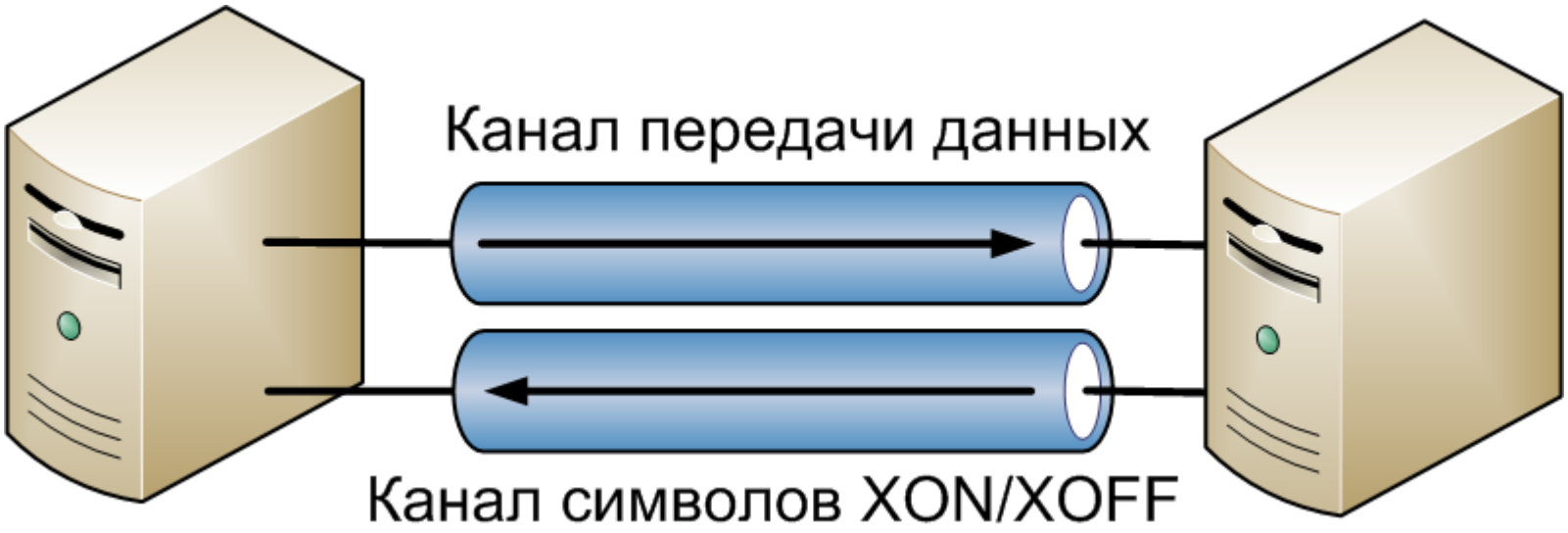


Шаг 2. Принимающая сторона присылает подтверждение приема

Эффективность использования канала передачи данных невысокая.

Проблема 2. Управление потоком информации

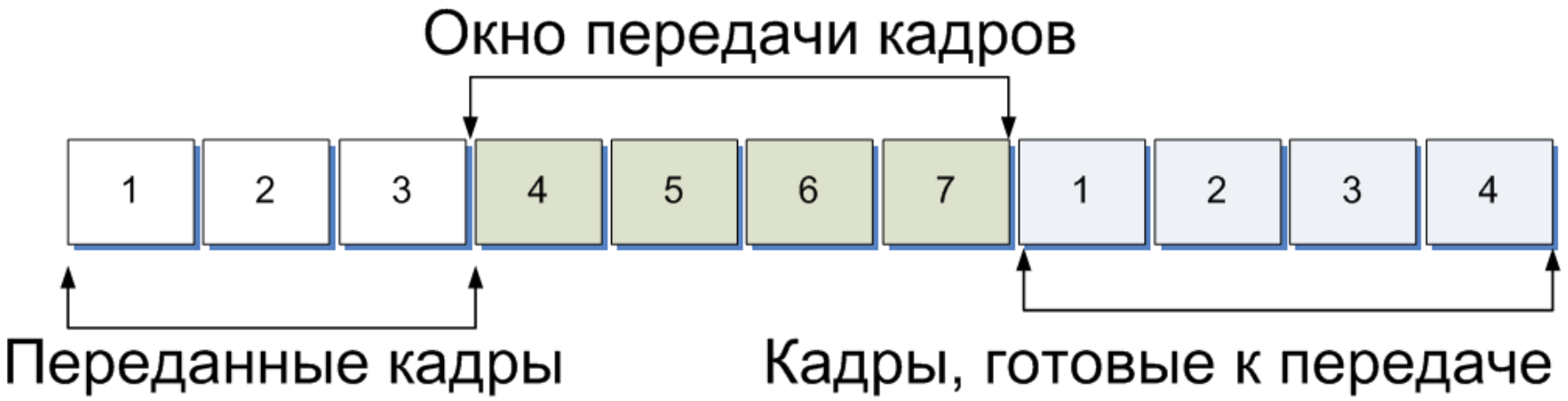
Способ решения проблемы 4 – Режим XON/XOFF



- Разновидность режима «Остановка»-«Ожидание» при наличии двунаправленного канала передачи данных.
- В принимающем устройстве необходимо наличие буфера и предсказательной отправки символа XOFF.
- XON = 11h, XOFF = 12h.

Проблема 2. Управление потоком информации

Способ решения проблемы 5 – Режим «скользящего окна»

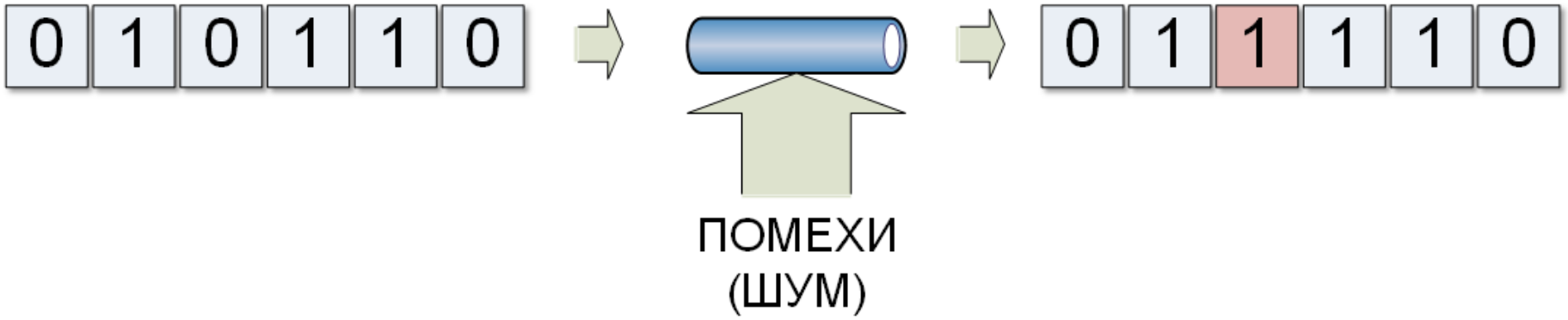


- Комбинирует достоинства режима «остановка» – «ожидание» и XON/XOFF.
- Кадры нумеруются (номер указывается в специальном поле кадра).
- Допускает запрос на повторную передачу кадра (кадров).

Проблема 2. Управление потоком информации

Способ решения проблемы 5 – Режим «скользящего окна» (продолжение)

Проблема 3. Определение ошибочно переданных сообщений

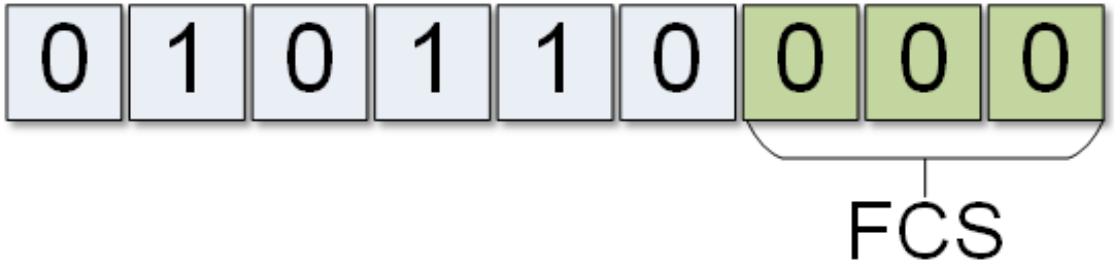


- Идеальных каналов связи не существует
- При передаче данные могут искажаться
- Необходимо на стороне приемника убедиться, что данные получены без искажения
- При необходимости данные следует запросить повторно

Общий подход к решению проблемы – формирование контрольной суммы кадра

Суть методов определения ошибок при передаче кадров – формирование и передача дополнительных полей, содержащих информацию о передаваемых данных.

Дополнительные поля называются **контрольной последовательностью** или **контрольной суммой** кадра (FCS, Frame Check Sequence)

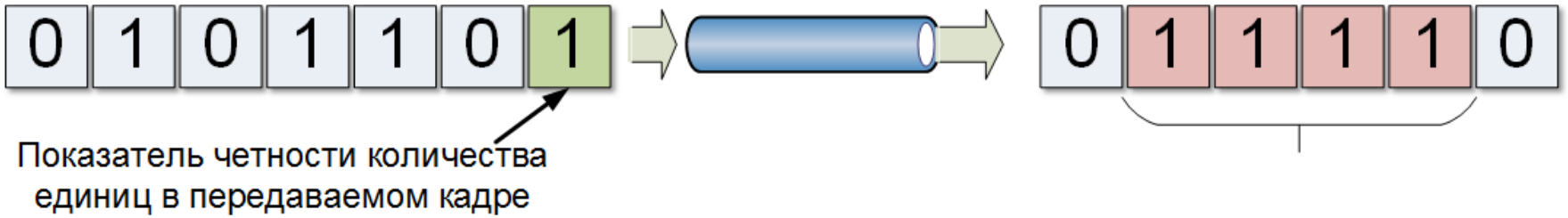


Неудобство – появление служебной информации (накладных расходов).

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 1. Контроль по паритету

Функция контрольной суммы по паритету = $x_1 \otimes x_2 \ldots \otimes x_n$, где $\otimes$ - сложение по модулю 2.



Метод контроля четности количества единиц в передаваемом кадре:

- Позволяет определить факт единичного (или нечетного) изменения количества разрядов в кадре
- Используется для передачи небольших кадров (RS-232, например).
- Известна модификация метода – матричный паритет

Видео про двоичную арифметику (отвлечемся ☺)

Функция контрольной суммы -
деление двух чисел по модулю 2 без учета переносов разрядов.

00110001001100100011001100110100001101010011011000110111001111000001110011

CRC32 =
11001011111110100001110010010011 0

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

Сложение с переносом

```
  11001010 = 202
+ 10011011 = 155
  ------------
 101100101 = 357
```

Сложение без переноса

```
  11001010 = 202
+ 10011011 = 155
  --------
  01010001 =  81
```

Результат – операция XOR.

Вычитание с переносом

```
  11001010 = 202
- 10011011 = 155
  ------------
  00101111 =  47
```

Вычитание без переноса

```
  11001010 = 202
- 10011011 = 155
  --------
  01010001 =  81
```

Результат – операция XOR.

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

```
          1100001010
        --------------
10011 ) 11010110110000
        10011.........
        -----.........
         10011........
         10011........
         -----........
          00001.......
          00000.......
          -----.......
           00010......
           00000......
           -----......
            00101.....
            00000.....
            -----.....
             01011....
             00000....
             -----....
              10110...
              10011...
              -----...
               01010..
               00000..
               -----..
                10100.
                10011.
                -----.
                  1110
```

Остаток

```
      x 1101
        1011
        ----
      + 1101
     + 1101.
    + 0000..
     1101...
     -------
     1111111
```

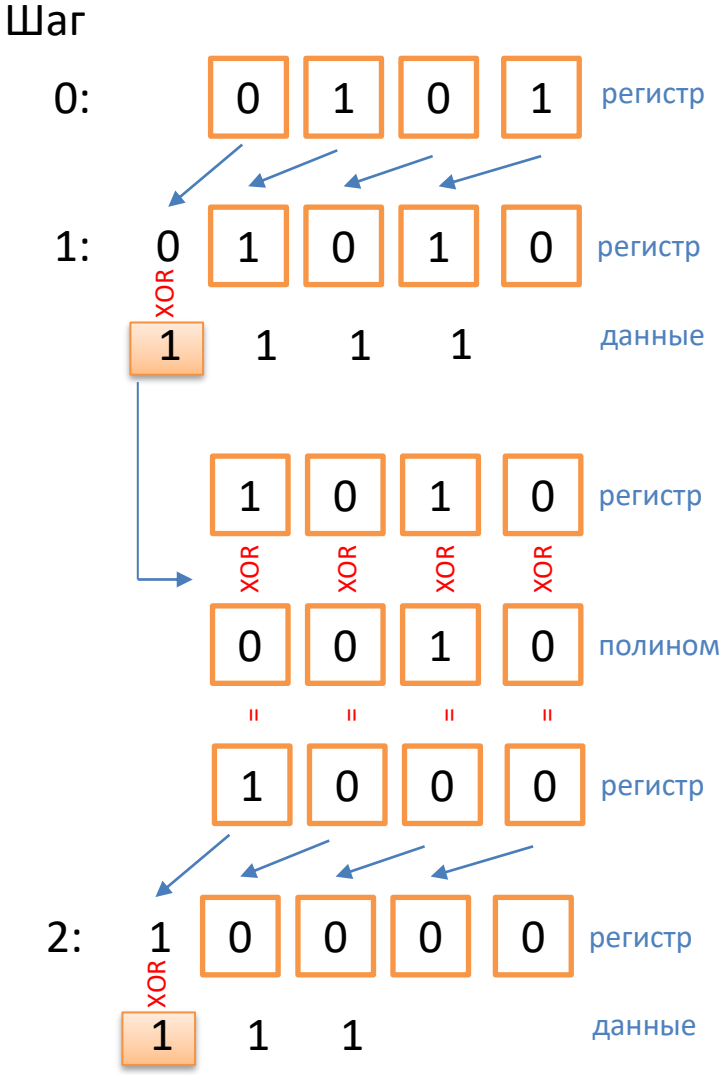
Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

```
В регистр загружается начальное значение
Сообщение выстраивается в заданном порядке и дополняется
нулями в количестве W (длинна остатка)
Пока (есть необработанные биты в сообщении)
  Регистр сдвигается на 1 разряд влево
  Если (в старшем разряде регистра до сдвига была 1 И
            очередной бит последовательности равен 1) Тогда
    Регистр = Регистр XOR Полином-делитель
  КонецЕсли
  Выбрать очередной бит из последовательности
КонецЦикла
Биты регистра выстраиваются в заданном порядке
Результат = Регистр XOR Конечное значение
```

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение



Заполняем регистр начальным значением

Выбираем первый разряд из кодируемой последовательности и делаем XOR со старшим разрядом регистра. Сдвигаем регистр на один разряд влево

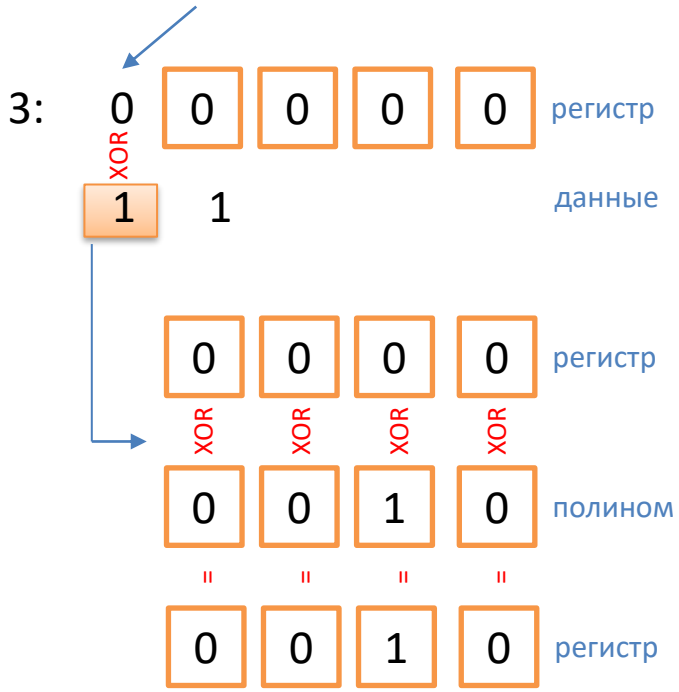
Если операция XOR с разрядами = 1, то делаем XOR между регистром и полиномом

Выбираем первый разряд из кодируемой последовательности и делаем XOR со старшим разрядом регистра. Сдвигаем регистр на один разряд влево

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

Шаг



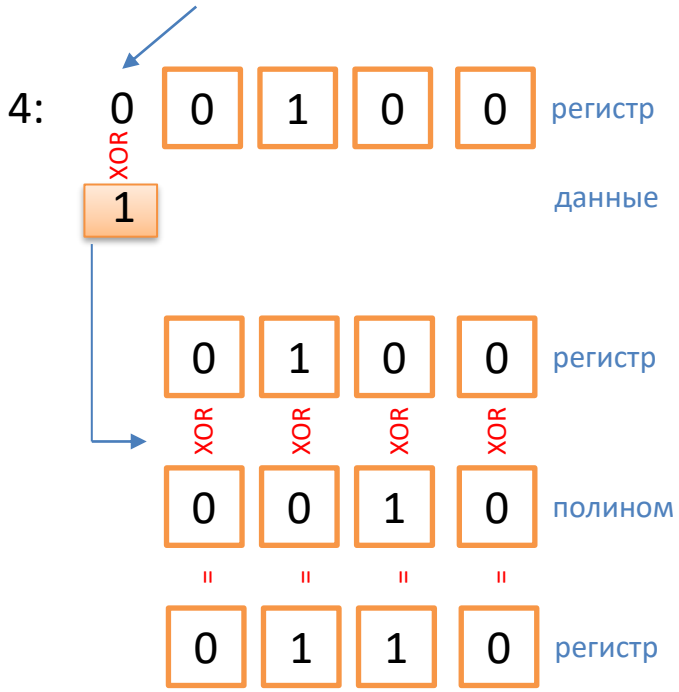
Выбираем первый разряд из кодируемой последовательности и делаем XOR со старшим разрядом регистра. Сдвигаем регистр на один разряд влево

Если операция XOR с разрядами = 1, то делаем XOR между регистром и полиномом

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

Шаг



Выбираем первый разряд из кодируемой последовательности и делаем XOR со старшим разрядом регистра. Сдвигаем регистр на один разряд влево

Если операция XOR с разрядами = 1, то делаем XOR между регистром и полиномом

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

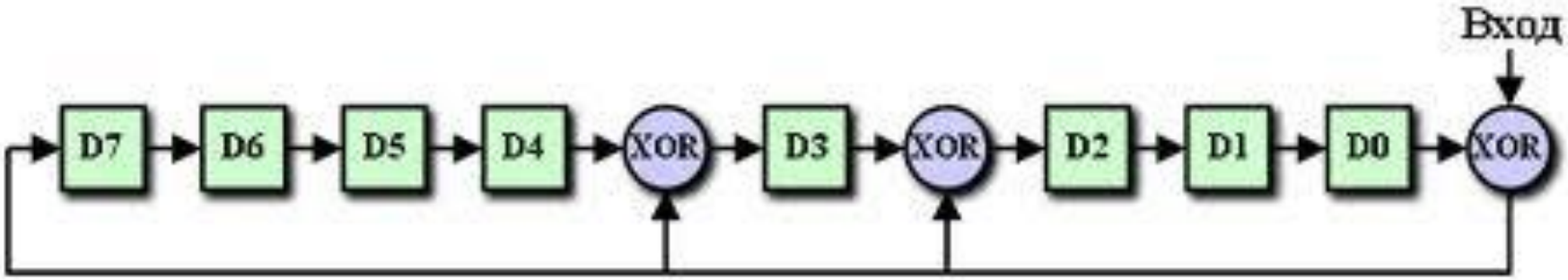


Схема формирования контрольной суммы CRC-8.
Порождающий многочлен $g(x) = x^8+x^5+x^4+1^1$

Картинка позаимствована из энциклопедии WiKi (https://ru.wikipedia.org/wiki/Циклический_избыточный_код)

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

Name: Имя алгоритма
Width: Длинна полинома-делителя
Poly: Значение полинома
Init: Начальное значение регистра
RefIn: Порядок просмотра бит в байтах входной последовательности
RefOut: Порядок байт в регистре с остатком от деления
XorOut: Выходное значение для суммирования
Check: Контрольное значение

Name: CRC32
Width: 32
Poly: 04C11DB7
Init: FFFFFFFF
RefIn: True
RefOut: True
XorOUT: FFFFFFFF
Check: CBF43926

Name: CRC16/CITT
Width: 16
Poly: 1021
Init: FFFF
RefIn: False
RefOut: False
XorOUT: 0000
Check: 29B1

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

```
char str[] = "123456789\0";
int crc = 0xFFFFFFFF, polynom = 0x04C11DB7, i = 0, j = 0;
char bit;


while (str[i] != '\0'){
  for (j = 0; j < 8; j++){
    bit = ((str[i] >> j) & 1) ^ ((crc >> 31) & 1);
    crc = crc << 1;
    if (bit) {
      crc = crc ^ polynom;
    }
  }
  i++;
}
crc = reflect (crc,32);
crc = crc ^ 0xFFFFFFFF;
```

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 2. Циклический избыточный код (CRC). Продолжение

```
char str[] = "123456789\0";
unsigned short int crc = 0xFFFF, polynom = 0x1021, i = 0, j = 0;
char bit;


while (str[i] != '\0'){
  for (j = 0; j < 8; j++){
    bit = ((str[i]>>(7-j))&1) ^ ((crc >> 15) & 1);
    crc = crc << 1;
    if (bit) {
      crc = crc ^ polynom;
    }
  }
  i++;
}
```

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 3. Корректирующий код Хемминга*

Исходная последовательность:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
1	0	0	1	0	0	1	0	1	1	1	0	0	0	1

Длина кодовой последовательности: $k = \log_2(k + n + 1)$

Кодовая последовательность передается вместе с исходной. Разряды кодовой последовательности помещаются в разряды с номерами 2^i:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_0	r_1	X_1	r_2	X_2	X_3	X_4	r_3	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	r_4	X_{12}	X_{13}	X_{14}	X_{15}
		1		0	0	1		0	0	1	0	1	1	1		0	0	0	1

Пример из базы Wiki http://ru.wikipedia.org/wiki/Код_Хэмминга

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 3. Корректирующий код Хемминга (продолжение)

Кодирование последовательности:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_0	r_1	X_1	r_2	X_2	X_3	X_4	r_3	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	r_4	X_{12}	X_{13}	X_{14}	X_{15}
1	0	1	1	0	0	1	0	0	0	1	0	1	1	1	1	0	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

$$r_i = \left(\sum_{j=1}^{n+k} z_j * y_{ij}\right) mod\ 2$$

z_i – кодируемая последовательность, с учетом вставляемых разрядов. yij – элементы расчетной таблицы. В процессе расчета в кодируемой последовательности $r_i = 0$.

Проблема 3. Определение ошибочно переданных сообщений

Способ решения 3. Корректирующий код Хемминга (продолжение)

Декодирование последовательности:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_0	r_1	X_1	r_2	X_2	X_3	X_4	r_3	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	r_4	X_{12}	X_{13}	X_{14}	X_{15}
1	0	1	1	0	0	1	0	1	0	1	0	1	1	1	1	0	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Полученная последовательность кодируется аналогично: R = (0, 0, 1, 1, 1)

Разница результатов формирует синдром: S = (1, 0, 0, 1, 0)

Синдром определяет изменившийся разряд

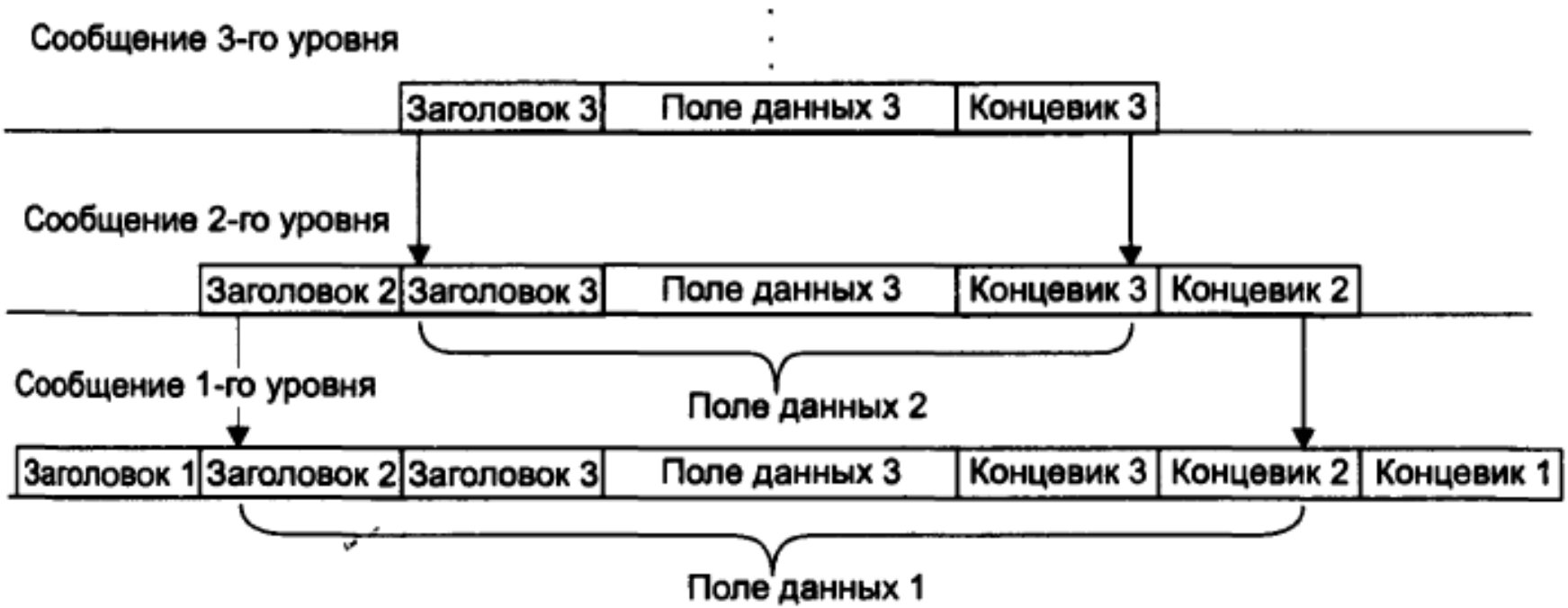
Проблема 3. Определение ошибочно переданных сообщений

Способ решения 3. Корректирующий код Хемминга (продолжение)

Правильное ли переданное сообщение?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r_0	r_1	X_1	r_2	X_2	X_3	X_4	r_3	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	r_4	X_{12}	X_{13}	X_{14}	X_{15}
1	1	0	1	1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Инкапсуляция:

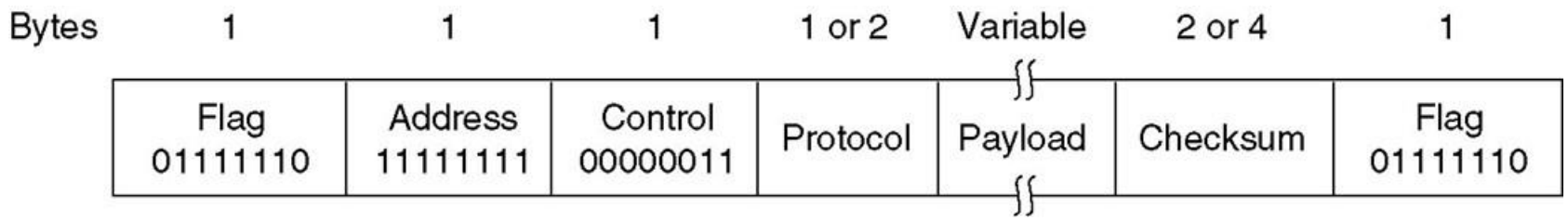


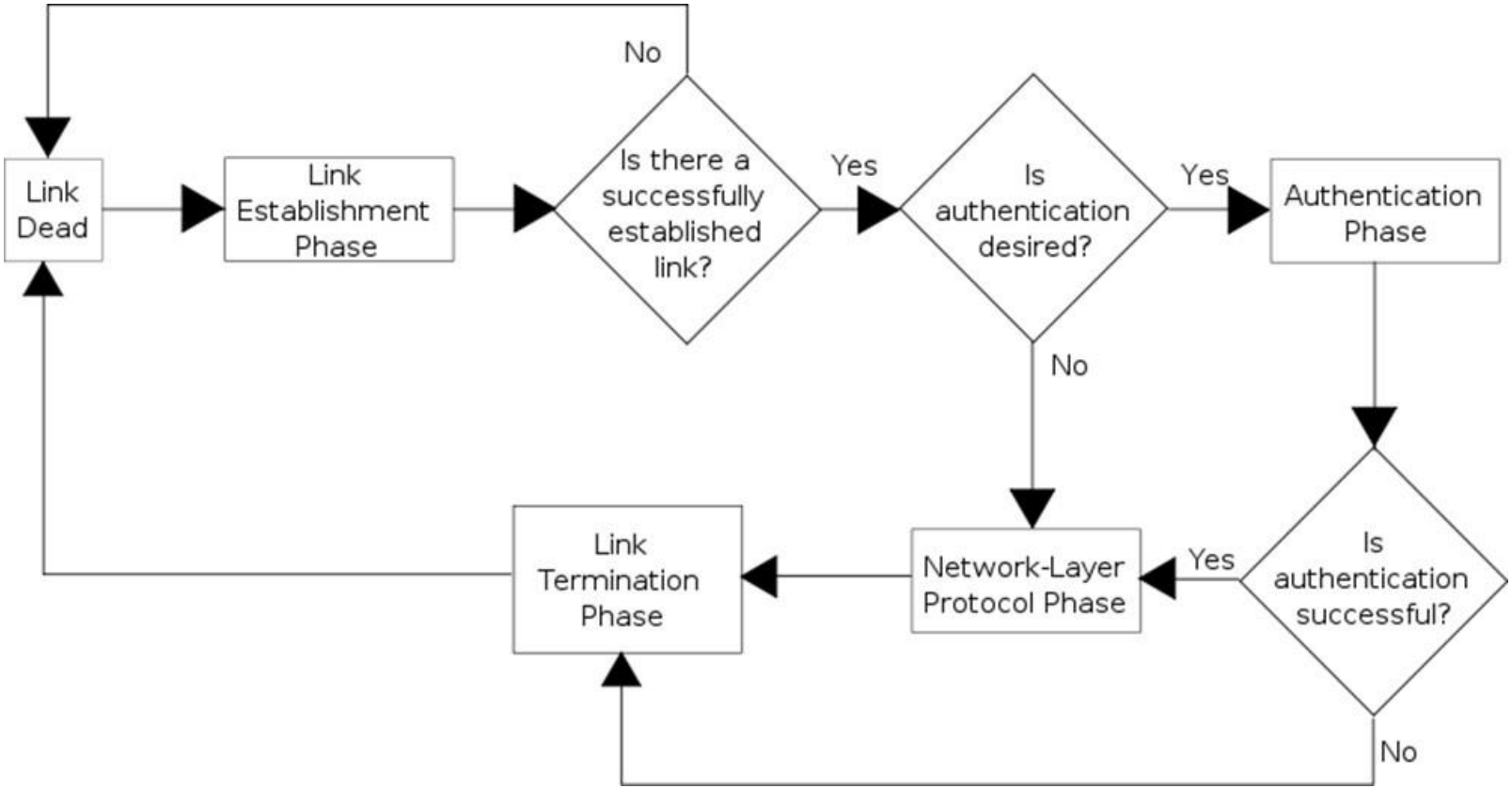
Протокол HDLC (англ. Higher-level Data Link Control)

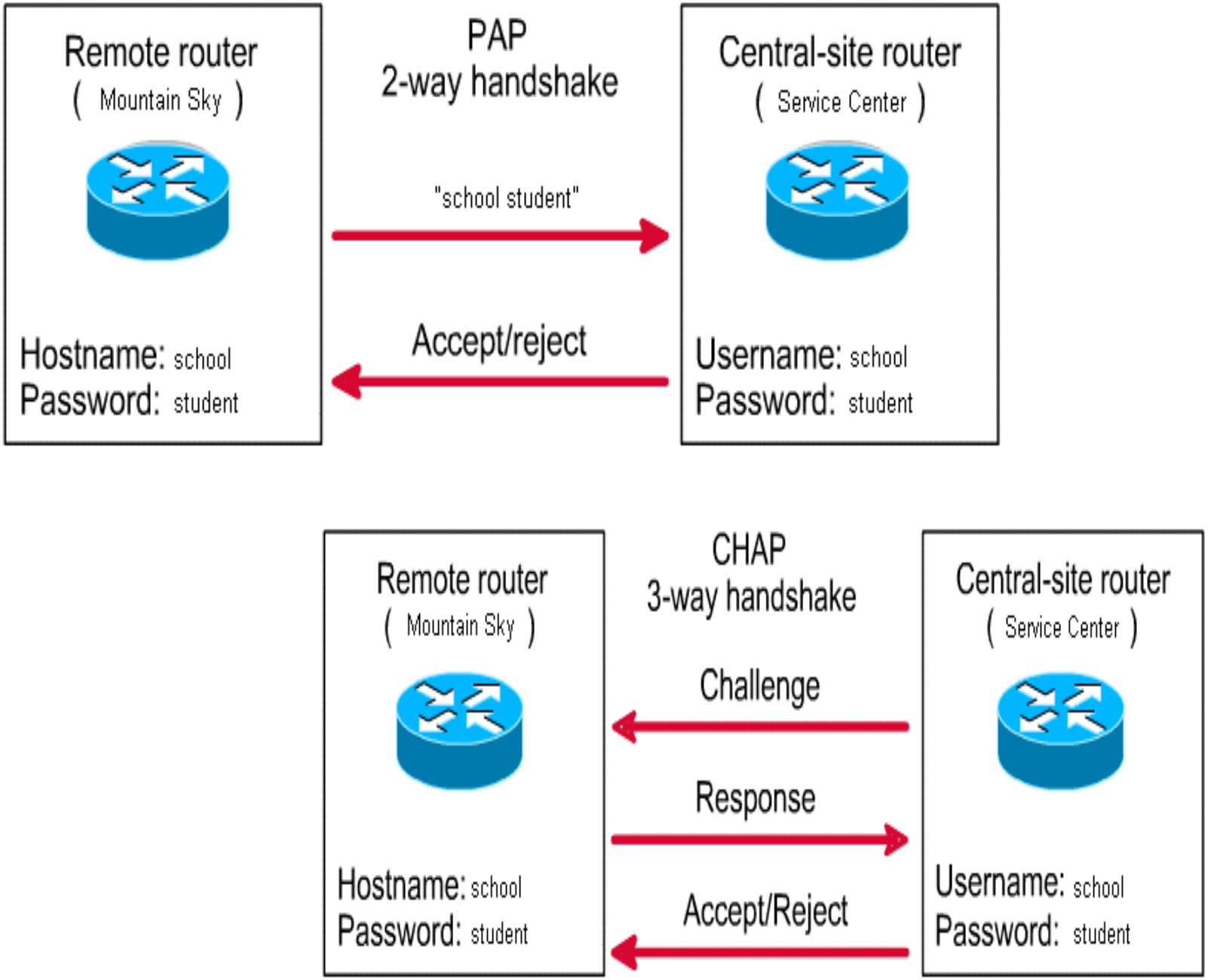
Формат кадра:

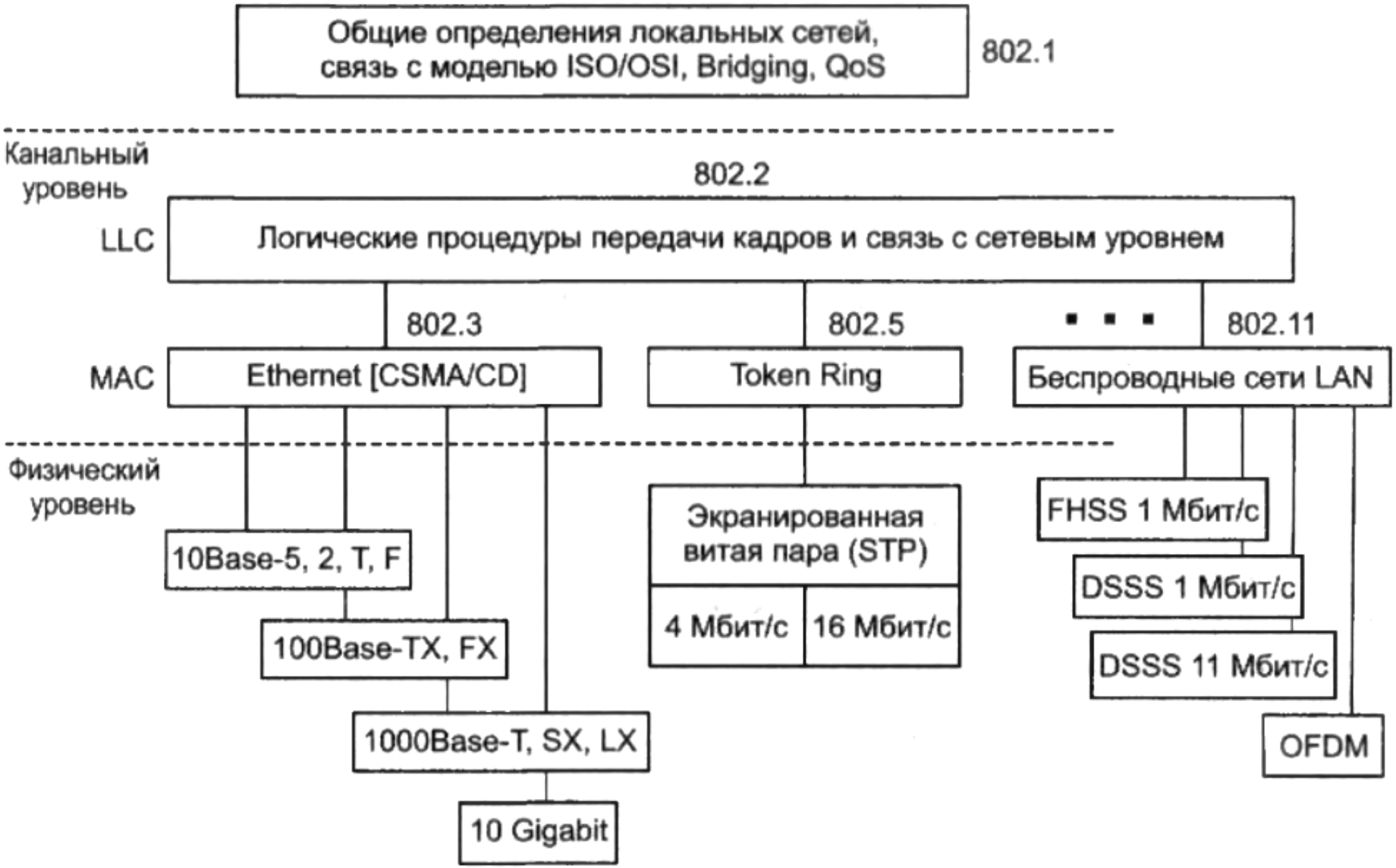
- FD (англ. Frame Delimiter) - разделители кадров = 0x7E.
  (используются для синхронизации приемника и передатчика. Бит- и байтстаффинг (5 единиц, 0)
- FCS (англ. Frame Check Sequence) – контрольная сумма кадра. CRC-16/CCIT

Протокол PPP (англ. Point-to-Point)






Remote router
(Mountain Sky)
Hostname: school
Password: student
PAP
2-way handshake
"school student"
Accept/reject
Central-site router
(Service Center)
Username: school
Password: student
Remote router
(Mountain Sky)
Hostname: school
Password: student
CHAP
3-way handshake
Challenge
Response
Accept/Reject
Central-site router
(Service Center)
Username: school
Password: student


Общие определения локальных сетей, связь с моделью ISO/OSI, Bridging, QoS
802.1
Канальный уровень
802.2
LLC
Логические процедуры передачи кадров и связь с сетевым уровнем
802.3
802.5
802.11
MAC
Ethernet [CSMA/CD]
Token Ring
Беспроводные сети LAN
Физический уровень
10Base-5, 2, T, F
100Base-TX, FX
1000Base-T, SX, LX
10 Gigabit
Экранированная витая пара (STP)
4 Мбит/с
16 Мбит/с
FHSS 1 Мбит/с
DSSS 1 Мбит/с
DSSS 11 Мбит/с
OFDM

Кадр 802.3/LLC

6	6	2	1	1	1(2)	46–1497 (1496)	4
DA	SA	L	DSAP	SSAP	Control	Data	FCS
			Заголовок LLC				

Кадр Raw 802.3/Novell 802.3

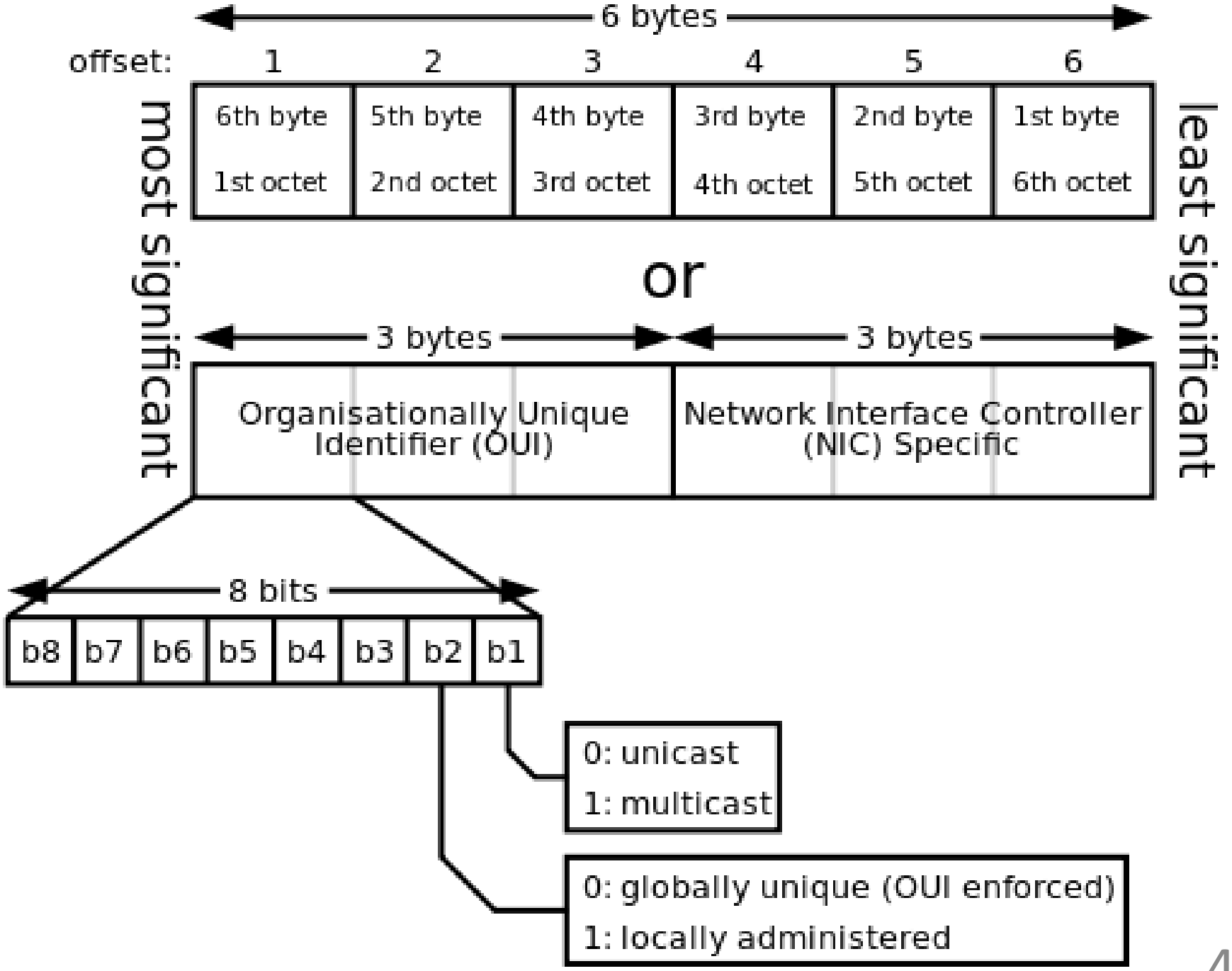
6	6	2	46–1500	4
DA	SA	L	Data	FCS

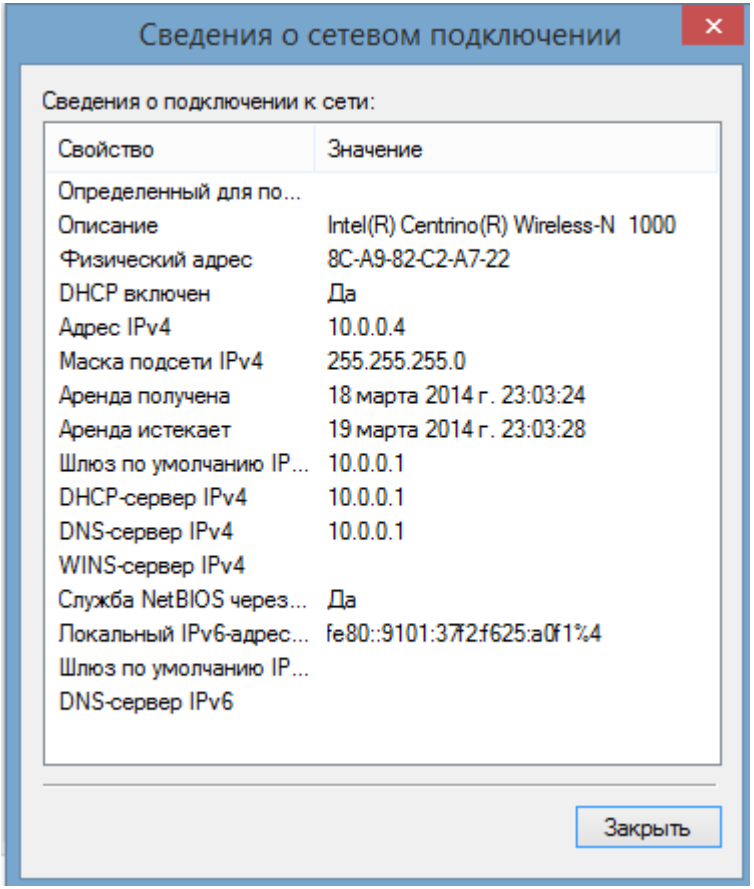
Кадр Ethernet DIX (II)

6	6	2	46–1500	4
DA	SA	T	Data	FCS

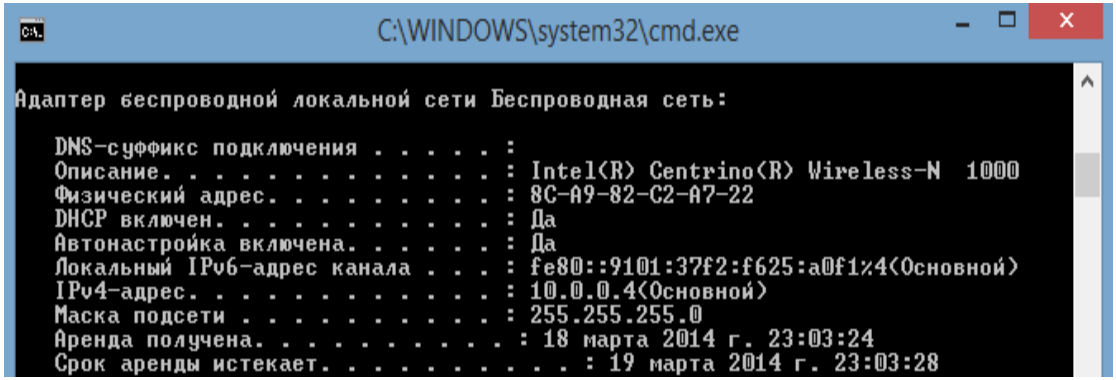
Кадр Ethernet SNAP

6	6	2	1	1	1	3	2	46–1492	4
DA	SA	L	DSAP	SSAP	Control	OUI	T	Data	FCS
			AA	AA	03	000000			
			Заголовок LLC			Заголовок SNAP			

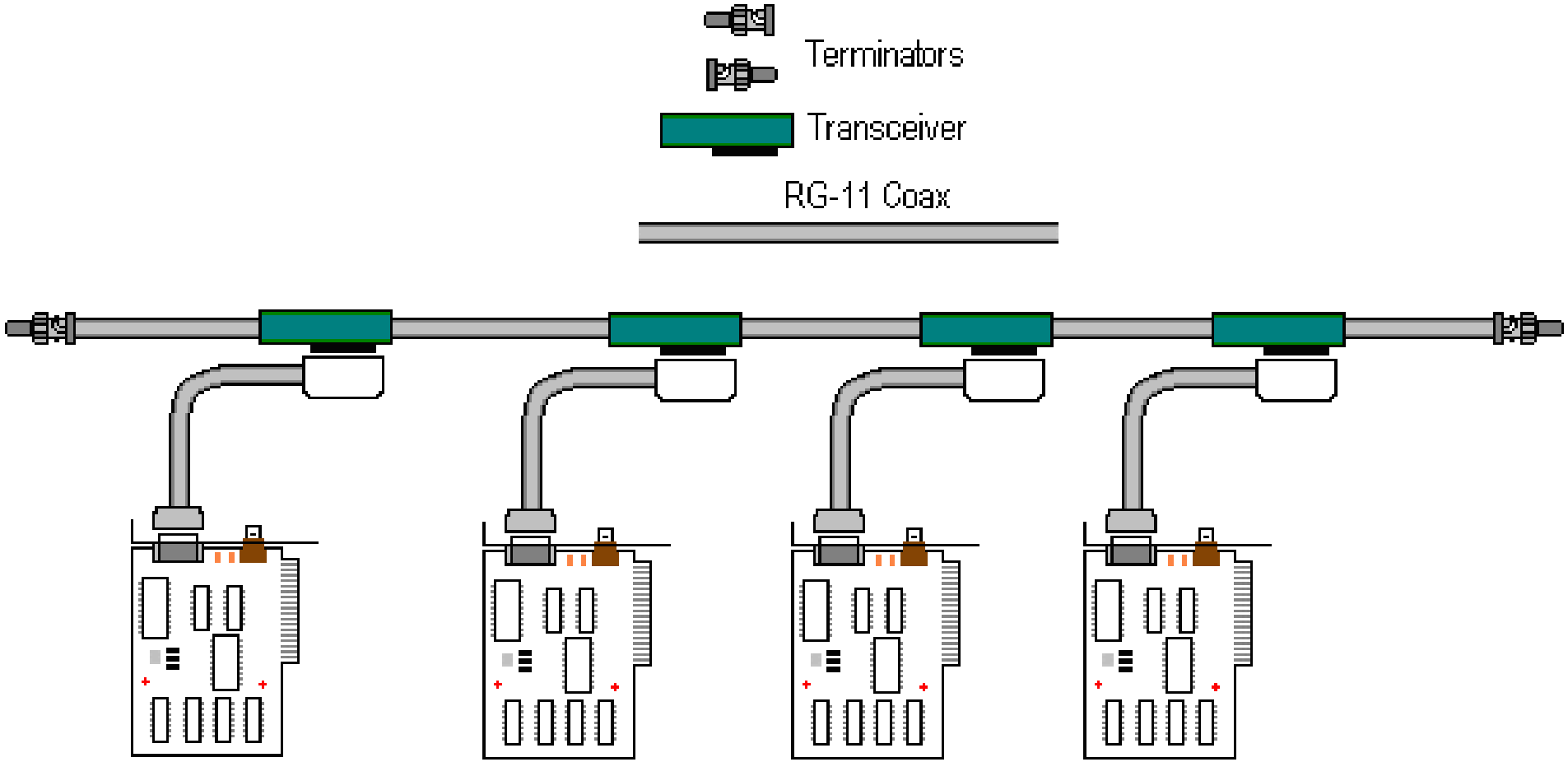

6 bytes
offset:
1
2
3
4
5
6
6th byte
1st octet
5th byte
2nd octet
4th byte
3rd octet
3rd byte
4th octet
2nd byte
5th octet
1st byte
6th octet
most significant
least significant
or
3 bytes
3 bytes
Organisationally Unique Identifier (OUI)
Network Interface Controller (NIC) Specific
8 bits
b8
b7
b6
b5
b4
b3
b2
b1
0: unicast
1: multicast
0: globally unique (OUI enforced)
1: locally administered

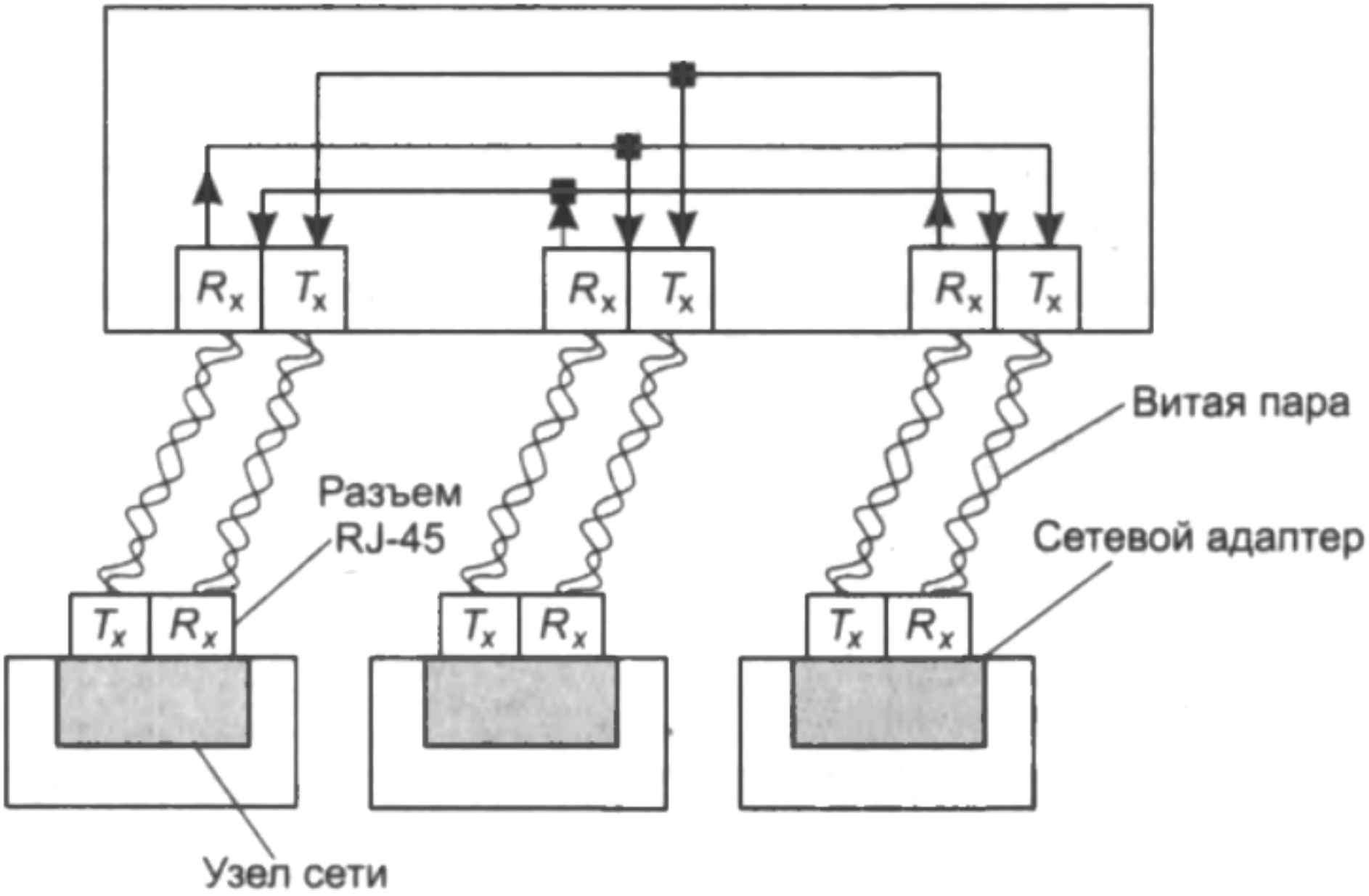



```
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state
    link/ether 00:22:15:d1:68:cd brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
en 1000
    link/ether 00:40:f4:70:4e:2e brd ff:ff:ff:ff:ff:ff
```




```
Switch>enable
Switch#show interfaces fastEthernet 0/1
FastEthernet0/1 is up, line protocol is up (connected)
  Hardware is Lance, address is 0002.165a.9901 (bia 0002.165a.9901)
```


Terminators
Transceiver
RG-11 Coax


Rx
Tx
Rx
Tx
Rx
Tx
Витая пара
Разъем
RJ-45
Сетевой адаптер
Tx
Rx
Tx
Rx
Tx
Rx
Узел сети

Ethernet: uses CSMA/CD

A: sense channel, **if** idle

then {

transmit and monitor the channel;

If detect another transmission

then {

abort and send jam signal;

update # collisions;

delay as required by exponential backoff algorithm;

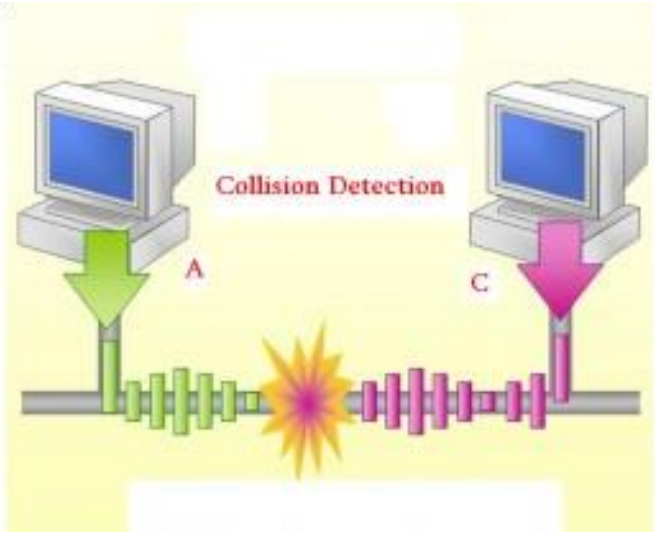
goto A

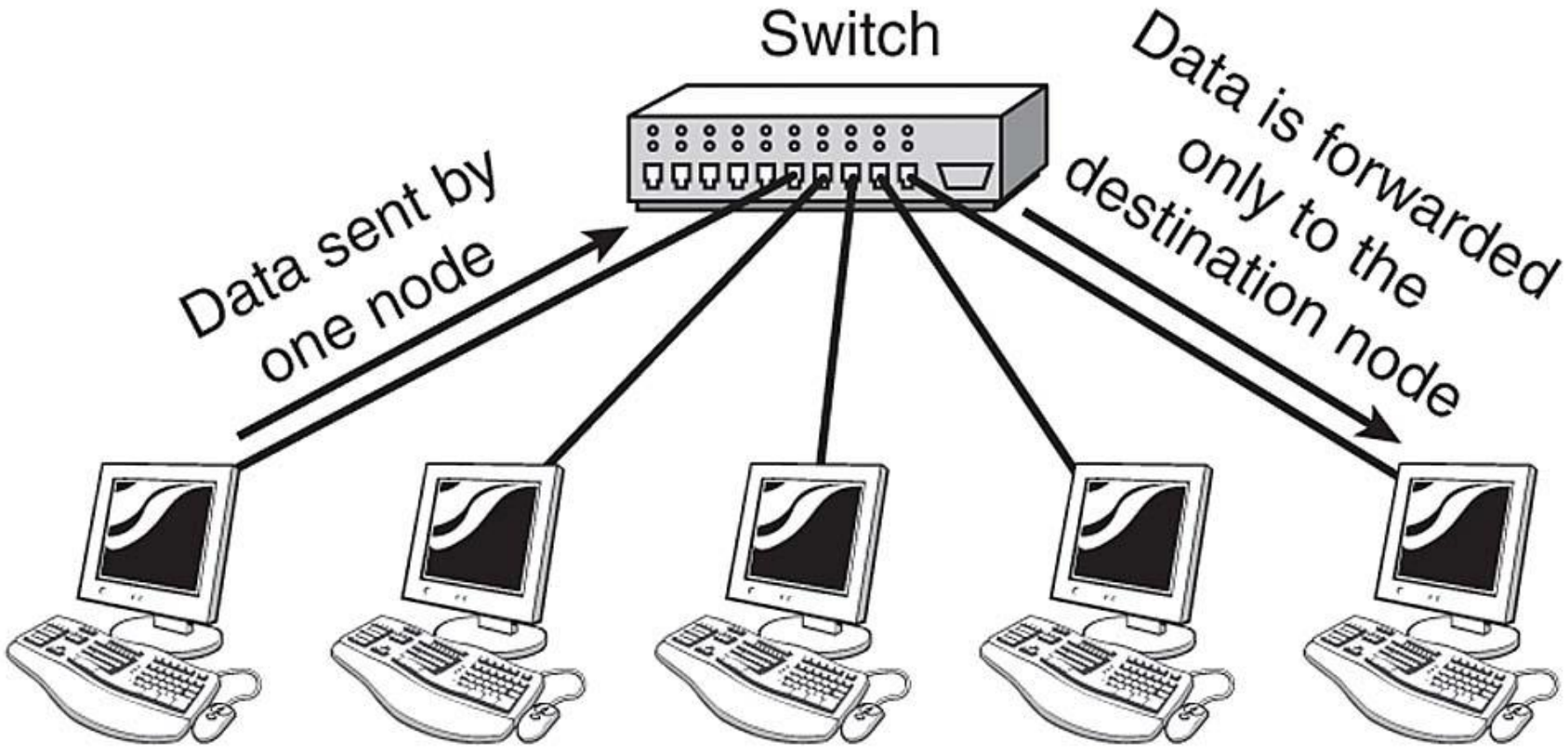
}

else {done with the frame; set collisions to zero}

}

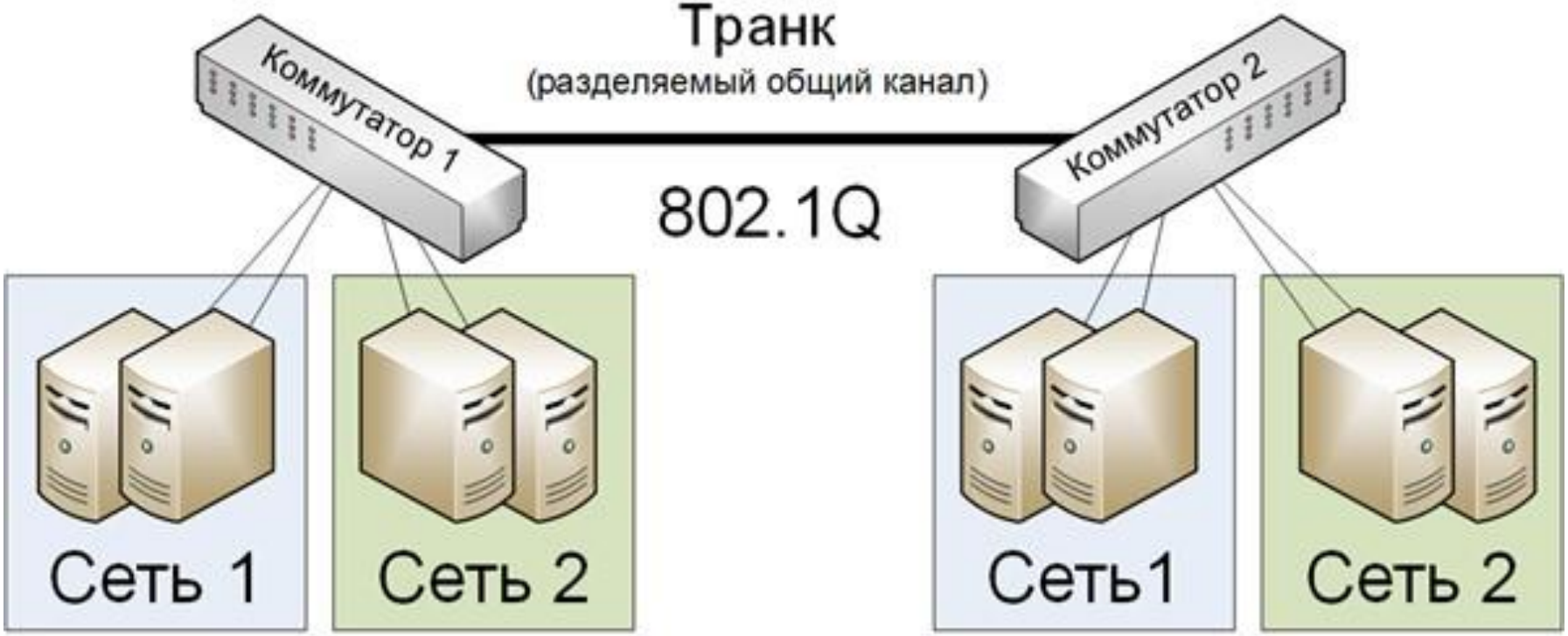
else {wait until ongoing transmission is over and goto *A*}

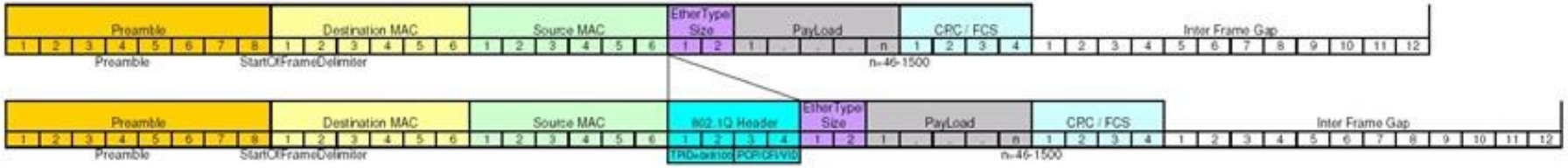




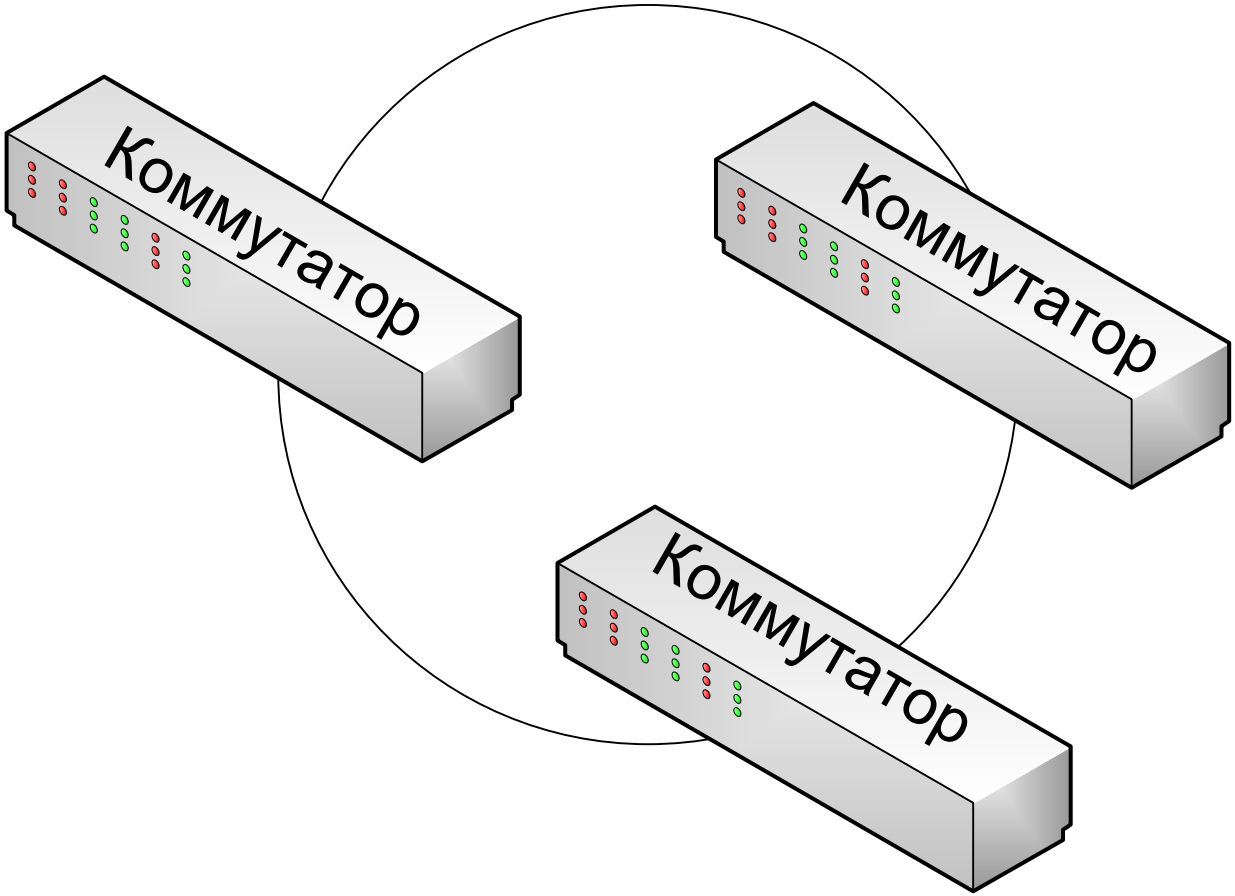
Технологии: Store-And-Forward, Cut-Through

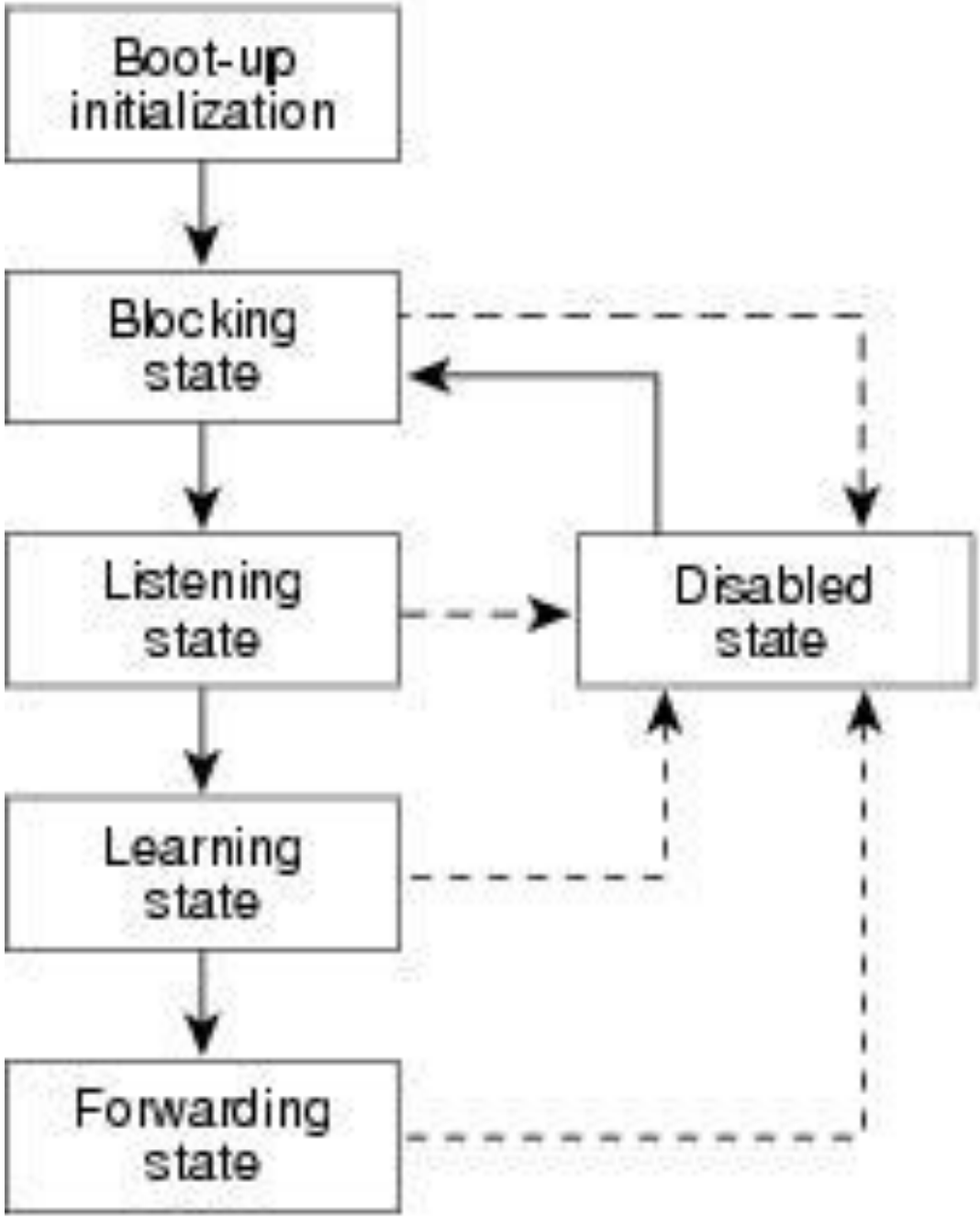
Домен коллизий


Транк
(разделяемый общий канал)
Коммутатор 1
Коммутатор 2
802.1Q
Сеть 1
Сеть 2
Сеть1
Сеть 2


Preamble
Destination MAC
Source MAC
EtherType/Size
PayLoad
CRC / FCS
Inter Frame Gap
Preamble
StartOfFrameDelimiter
n=46-1500
Preamble
Destination MAC
Source MAC
802.1Q Header
EtherType/Size
PayLoad
CRC / FCS
Inter Frame Gap
Preamble
StartOfFrameDelimiter
TPID=0x8100
PCP/CFI/VID
n=46-1500

Алгоритм построения дерева без колец (Spanning-Tree Protocol)



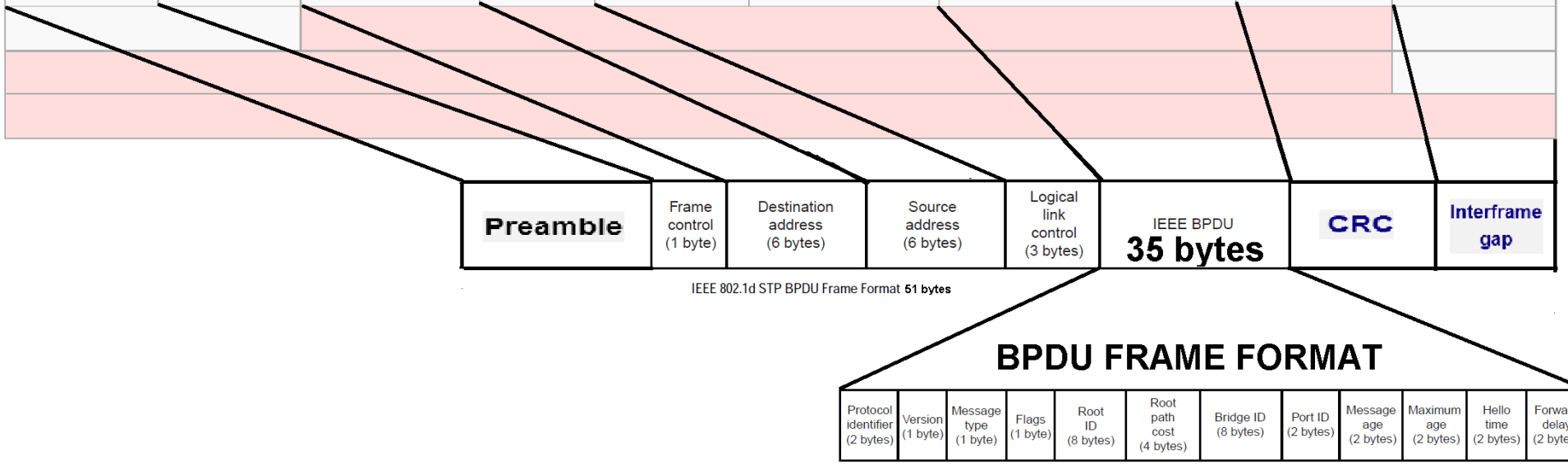

Boot-up initialization
Blocking state
Listening state
Disabled state
Learning state
Forwarding state
20 секунд
15 секунд
15 секунд

Фрейм BPDU имеет следующие поля:

- Идентификатор версии протокола STA (2 байта). Коммутаторы должны поддерживать одну и ту же версию протокола STA;
- Версия протокола STP (1 байт);
- Тип BPDU (1 байт). Существует 2 типа BPDU - конфигурационный и уведомление о реконфигурации;
- Флаги (1 байт);
- Идентификатор корневого коммутатора (8 байт);
- Расстояние до корневого коммутатора (4 байта);
- Идентификатор коммутатора (8 байт);
- Идентификатор порта (2 байта);
- Время жизни сообщения (2 байта). Измеряется в единицах по 0.5 сек, служит для выявления устаревших сообщений;
- Максимальное время жизни сообщения (2 байта). Если кадр BPDU имеет время жизни, превышающее максимальное, то кадр игнорируется коммутаторами;
- Интервал hello (2 байт), интервал через который посылаются пакеты BPDU;
- Задержка смены состояний (2 байта). Минимальное время перехода коммутатора в активное состояние;

802.3 Ethernet frame structure

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32–bit CRC)	Interframe gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42[note 2]–1500 octets	4 octets	12 octets



IEEE 802.1d STP BPDU Frame Format **51 bytes**

LACP 802.3ad