2 ЛАБОРАТОРНАЯ РАБОТА № 2. ПАКЕТНАЯ ПЕРЕДАЧА

2.1 Каналы передачи данных

Исторически так сложилось, что компьютерные сети имеют последовательную природу. Объяснить это можно тем, что реализовать передачу данных на сравнительно большие расстояния в параллельном виде значительно сложнее, чем в последовательном. Между станциями данные передаются по последовательным каналам, а внутри станций обрабатываются параллельно.

Последовательный канал может быть:

1. *Выделенным* (leased) – зарезервирован за определенной парой станций-абонентов (например, при топологии точка к точке).
2. *Разделяемым* (shared) – может использоваться несколькими станциями-абонентами (например, при шинной топологии).

Причем канал, который не может разделяться несколькими станциями-передатчиками одновременно, в отечественной литературе принято называть *моноканалом*. Во многих реализациях ситуация именно такая.

Отдельно выделяется и понятие *коммутируемого* канала – получаемого посредством коммутации разрозненных «каналов-частей».

Последовательный канал может функционировать в одном из трех режимов:

1. *Симплексном* (simplex) – передача данных по каналу возможна только в одном направлении.
2. *Полудуплексном* (semiduplex) – данные могут передаваться в обоих направлениях, но в один момент времени возможна передача только в одном направлении.
3. *Полнодуплексном* (full duplex) – данные могут передаваться в обоих направлениях одновременно.

UART 16550 работает в полнодуплексном режиме. В имеющейся лабораторной установке адаптер RS-485 функционирует в полудуплексном режиме. Для полноценной же программной реализации полнодуплексного режима необходимо наличие многопроцессорной системы.

2.2 Пакеты

Для именования порции информации, передаваемой по каналам компьютерных и не только компьютерных сетей, используется обобщенный термин *пакет* (packet). Пакет содержит последовательно сформированные станцией-передатчиком *поля* (fields), предназначенные для их интерпретации в станции-приемнике. В общем случае, пакеты могут быть самыми разнообразными (как по структуре, так и по размеру), но подавляющее большинство пакетов подпадают под типовую структуру (рисунок 2.1).

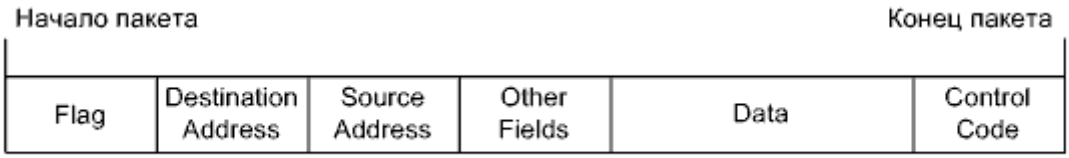

Рисунок 2.1 – Структура типового пакета компьютерной сети

Назначение полей:

1. Flag (флаг, точнее, флаг начала пакета) – позволяет определить начало пакета.
2. Destination Address (адрес назначения) – позволяет указать станцию, для которой предназначен пакет.
3. Source Address (адрес источника) – позволяет указать станцию, сгенерировавшую пакет.
4. Other Fields (прочие поля) – специфические поля (в том числе и специфические флаги) определенной реализации.
5. Data (данные) – «полезное» наполнение пакета.
6. Control Code (контрольная сумма) – позволяет проверить целостность пакета.

Часть пакета, включающую поля, расположенные до начала данных, принято называть *заголовком* (header) пакета.

Обычно в байт-ориентированных реализациях размер пакета кратен восьми битам, то есть пакет состоит из так называемых *октетов* (octets).

2.3 Инкапсуляция

2.4 Байт-стаффинг и бит-стаффинг

Понятно, что для правильной интерпретации пакета нужно его считать из канала полностью, причем с соблюдением последовательности. Если бы взаимодействующие станции работали бесконечно и находились в соответствующей степени готовности, то это не составляло бы особого труда. Но, поскольку станция-приемник может подключиться к каналу (да и вообще начать работать) в произвольный момент времени, возникает проблема, связанная с распознаванием флага начала пакета. Флаг начала пакета представляет собой зарезервированную цифровую последовательность, которая собственно позволяет станции-приемнику определить начало пакета. Проблема заключается в том, что такая же последовательность вполне может встретиться в пакете и после флага начала. Следовательно, возникает задача обеспечения уникальности флага начала пакета, то есть исключения этой последовательности из оставшейся части пакета.

Это достигается за счет действия, заключающегося в модификации следующей за флагом цифровой последовательности, которое в бит-

ориентированных системах называется *бит-стаффингом* (Bit Stuffing), а в байт-ориентированных – *байт-стаффингом* (Byte Stuffing).

При бит-стаффинге совпадающая с флагом последовательность разбивается с помощью вставки дополнительно бита с соответствующим значением. Применение бит-стаффинга приводит к увеличению размера пакета. С целью уменьшения связанных с бит-стаффингом «издержек», следует стремиться к минимизации количества вставок: разбивающий бит нужно вставлять после наиболее длинной уникальной подпоследовательности в флаговой последовательности.

Классическим флагом начала пакета является байт со значением 01111110b (7Eh). Оптимальная реализация бит-стаффинга при использовании классического флага проиллюстрирована на рисунке 2.2.

Рисунок 2.2 – Пример реализации бит-стаффинга

На передающей стороне после нуля и шести единиц всегда вставляется седьмая единица, а на приемной стороне единица после нуля и шести единиц всегда удаляется.

Цель байт-стаффинга полностью совпадает с целью бит-стаффинга. В сравнении с алгоритмами бит-стаффинга, алгоритмы байт-стаффинга являются более сложными и более «затратными», но при программировании они позволяют избежать битовых операций.

На рисунке 2.3 показан один из вариантов реализации байт-стаффинга при использовании классического флага.

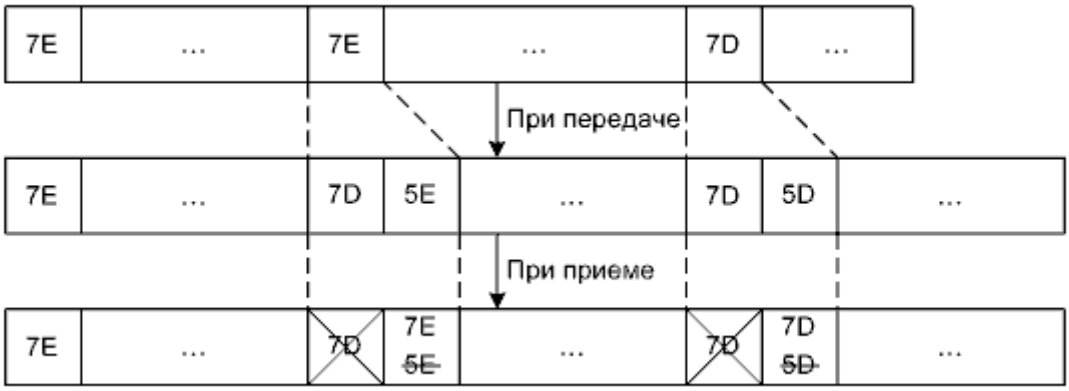

Рисунок 2.3 – Пример реализации байт-стаффинга

Единственным способом обеспечения уникальности флагового байта является замена совпадающего с ним байта на некий выбранный другой. Но возникает вопрос, как принимающая сторона отличит замененный байт от такого же незамененного. Решением является применение так называемого *ESC-символа*. Наличие ESC-символа говорит станции-приемнику о факте замены, а следующий за ESC-символом символ – код замены позволяет определить какая замена была осуществлена.

2.5 Практический пример