Учреждение образования

Белорусский государственный университет

информатики и радиоэлектроники

Кафедра электронных вычислительных машин

Лабораторная работа №2

**ПАКЕТНАЯ ПЕРЕДАЧА ДАННЫХ**

Выполнил: Проверил:

ст. гр. №750502 Глоба А. А.

Альховик Д. Г.

Минск, 2019

**1**. **Пакеты данных.**

Для именования порции информации, передаваемой по каналам компьютерных (и не только компьютерных) сетей, используют 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 длине), но подавляющее большинство пакетов подпадают под типовую структуру.



Рисунок 1 - Структура типового пакета КС

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. FCS (Frame Check Sequence) -- контрольная сумма -- позволяет проверить

целостность пакета.

Часть пакета, включающую поля, расположенные до начала данных, принято

называть *заголовком* (header) пакета, после данных -- *хвостовиком* (trailer).

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

Все поля в составе любого пакета можно условно разделить на полезные и

служебные. *Полезная нагрузка* (payload) заключается в собственно данных. Но

следует понимать, что вкладываемая в качестве данных информация может носить

служебный характер. В некоторых пакетах поле данных не предусмотрено вообще. Сколько дополнительного трафика порождается в связи с наличием служебных полей оценивают как overhead.

**2**. **Передача пакетов через уровни OSI.**

В соответствии с концепцией модели OSI, соседние уровни абстрагированы друг от друга. Поэтому вполне закономерно, что на каждом уровне работают со своими

структурами данных. При продвижении информации между уровнями возникает

необходимость в преобразованиях структур данных. Преобразования выражаются в инкапсуляции и декапсуляции. Под *инкапсуляцией* (encapsulation) в КС понимают вкладывание пакета определенного вышестоящего уровня в поле данных пакета смежного нижестоящего уровня в процессе подготовки к передаче, то есть при продвижении сверху вниз. Под *декапсуляцией* (decapsulation) понимают обратное действие после приема, то есть при продвижении снизу вверх.



Рисунок 2 - Инкапсуляция пакетов

Функционал любого из вышестоящих уровней «знает», какие нижестоящие ресурсы ему необходимы и чем он «располагает». Поэтому процесс инкапсуляции не доставляет трудностей. А вот функционал нижестоящего уровня при разборе

полученных пакетов заранее не знает, какой из вышестоящих подсистем передавать эти пакеты. Проблему решают введением в структуру пакета служебного поля, в котором записывается код протокола вышестоящего уровня.

Важной особенностью инкапсуляции является то, что в большинство реализаций

заложена возможность передавать пакеты, относящиеся к некоторому протоколу

некоторого уровня (например, сетевого), вкладывая их в пакеты другого протокола

того же уровня, то есть организовывать *туннелирование* (tunneling).

Если при выполнении инкапсуляции данные некоторого уровня не помещаются в

поле отведенной длины, то можно прибегнуть к фрагментации (fragmentation) --

разбить данные на фрагменты и передать цепочку пакетов. Принимающая сторона

будет вынуждена выполнить *дефрагментацию* (defragmentation).

*Перемежение* (interleaving) позволяет «распараллелить» пересылку пакетов или их

фрагментов и заключается в одновременном задействовании нескольких каналов.

Особенно это применимо в низкоскоростных СрПД.

Названия структурных единиц передаваемой информации в привязке к уровням

модели OSI:

L1 -- сигналы (signals).

L2 -- кадры (frames).

L3 -- собственно пакеты (packets).

L4 + L5 -- сегменты (segments).

L6 + L7 -- сообщения (messages).

**3. Бит- и байт- 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

называют *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 длины пакета. Теоретически, 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).

****

Рисунок 3 - Пример реализации бит-стаффинга

Цель байт-стаффинга полностью совпадает с целью бит-стаффинга. В сравнении с

алгоритмами бит-стаффинга, алгоритмы байт-стаффинга манипулируют байтами,

являются более сложными и более «затратными», но при программировании они

позволяют избежать битовых операций (бит-стаффинг, в отличие от байт-стаффинга, обычно реализуется аппаратно).

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. Байт-стаффингу можно подвергать целые группы символов.



Рисунок 4 - Пример реализации байт-стаффинга

**Вывод:**

Информация в последовательном канале передаётся порциями – пакетами. Пакеты реализуют не только более удобную и быструю передачу данных, но и помогают в проверке и исправлении переданной информации.

Для правильной интерпретации пакета нужно его считать из канала полностью, причем с соблюдением последовательности, так как пакеты разделяются специальными флагами. Однако при такой передаче может возникнуть проблема присутствия такой последовательности битов в полезной нагрузке. Для решения этой проблемы используются бит- или байт- стаффинг, который помогает так преобразовать полезную нагрузку, чтобы флаги не появлялись в ней.