БГУИР

Кафедра ЭВМ

Лабораторная работа №2

Пакетная передача данных

Выполнил Проверил:

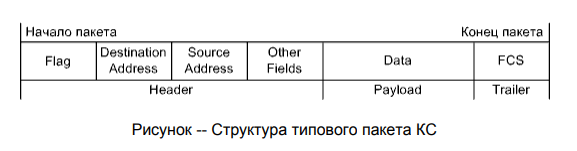
Студент группы 750502 Глоба А.А

Никанов И.В.

Минск 2019

**Основные теоретические сведения:**

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.

****

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.

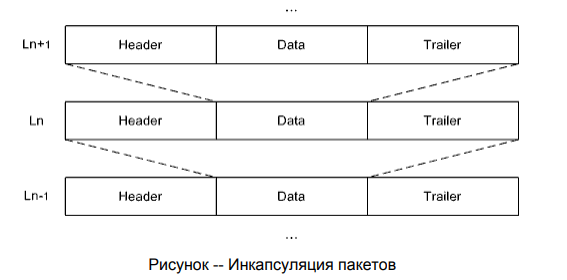
5. Data -- данные -- «полезное» наполнение пакета.

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).

Все поля в составе любого пакета можно условно разделить на полезные и служебные. Полезная нагрузка (payload) заключается в собственно данных.

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.

****

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 -- байтстаффингом (byte stuffing).

При бит-стаффинге совпадающая с флагом последовательность разбивается с помощью вставки дополнительно бита с соответствующим значением.

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 обычно применяется при задействовании синхронных СрПД, а байтстаффинг -- асинхронных.

**Пример передачи данных используя байт-стаффинг в лабораторной работе:**

7E – флаг начала пакета

7D – ESC-символ

5E – код замены флага начала пакета

5D – END-символ

Передаваемые данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| … | 7E | … | 7D | … |

При передаче(Encode):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7E | FF | 00 | … | 7D | 5E | … | 7D | 7D | … | 5D |

При приёме(Decode):

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **~~7~~**~~E~~ | ~~FF~~ | ~~00~~ | … | ~~7D~~ | 7E | … | 7D | ~~7D~~ | … | ~~5D~~ |
| ~~5E~~ |

Алгоритм “запаковки” заключается в следующем:

1) Сначала устанавливаем флаг начала пакета;

2) Если найден ESC-символ, то следующим за ним ставим END-символ;

3) Если найден символ, совпадающий с флагом начала, то заменяем его на ESC-символ и добавляем после него код замены.

Алгоритм распаковки следующий:

1) Если найден ESC-символ и за ним следует код замены, то заменяем ESC-символ на флаг начала и удаляем код замены;

2) Если найден END-символ, то удаляем его.

**Код лабораторной работы:**

Файл ByteStuffing.java:

public class ByteStuffing {  
  
 //first and last bytes are  
public final static byte *F\_END\_SYMBOL* = 0x7E; //126 ~, флаг начала пакета  
public final static byte *F\_ESC\_SYMBOL* = 0x7D; //125 }, ESC-символ  
public final static byte *T\_END\_SYMBOL* = 0x5E; //94, код замены начала пакета  
public final static byte *T\_ESC\_SYMBOL* = 0x5D; //93, END-символ

public static byte[] doStuffing(byte[] bytes) {  
  
 ArrayList<Byte> list = new ArrayList<>  
 (Arrays.*asList* (ArrayUtils.*toObject*(bytes)));  
  
 list.add(0, *F\_END\_SYMBOL*); //устанавливаем флаг начала пакета  
  
 for (int i = 1; i < list.size(); i++) {  
 list.trimToSize();  
 if (list.get(i) == *F\_ESC\_SYMBOL*) { //если найден ESC-символ,  
 list.add(i+1, *T\_ESC\_SYMBOL*); //то следующим за ним ставим END-символ  
 //i++;}  
 else if (list.get(i) == *F\_END\_SYMBOL*) { //если найден символ, совпадающий с флагом начала,  
 list.set(i, *F\_ESC\_SYMBOL*); //то заменяем его на ESC-символ  
 list.add(i+1, *T\_END\_SYMBOL*); //и добавляем после него код замены  
 //i++;}}  
  
 Byte[] returnBytes = new Byte[list.size()];  
 list.toArray(returnBytes);  
  
 System.*out*.println(ArrayUtils.*toPrimitive*(returnBytes));  
  
 return ArrayUtils.*toPrimitive*(returnBytes);  
}  
  
 public static byte[] inject(byte[] bytes) {  
 ArrayList<Byte> list = new ArrayList<>  
 (Arrays.*asList* (ArrayUtils.*toObject*(bytes)));  
 list.remove(0);  
 for (int i = 0; i < list.size(); i++) {  
 list.trimToSize();  
 if (list.get(i) == *F\_ESC\_SYMBOL*) { //если найден ESC-символ  
 if (list.get(i + 1) == *T\_END\_SYMBOL*) { //и за ним следует код замены,  
 list.set(i, *F\_END\_SYMBOL*); //то заменяем ESC-символ на флаг начала  
 list.remove(i+1); //и удаляем код замены  
 }  
 else if (list.get(i+1) == *T\_ESC\_SYMBOL*) { //если найден END-символ,  
 list.remove(i + 1);//то удаляем его

//i++;}}}  
 Byte[] returnBytes = new Byte[list.size()];  
 list.toArray(returnBytes);  
 return ArrayUtils.*toPrimitive*(returnBytes);  
 }}