Asian Journal of Information Technology 5 (1) : 65-68, 2006

A New Encryption Method for Short Message Service (SMS)

[1]Mohammad Abu Yousuf, [2]Mustafa Hasan, [3]S.M. Saif Shams
[1,2,3]Department of Computer Science and Engineering, [1]Mawlana Bhashani Science and Technology University, [2]Premier University, [3]Shahjalal University of Science and Technology ,[1]Santosh, Tangail-1902, [2]Chittagong, [3]Sylhet-3114 Bangladesh

Abstract: This study shows an encryption method for Short Message Service (SMS) to reduce amount of required memory space of data for device or to increase the number of bytes to transmit. SMS is a very popular service. SMS are developing very rapidly throughout the world. As SMS is an offline message service it's a huge pressure for server memory. In traditional SMS system it is possible to send maximum 160 characters and each character occupies one byte of memory space. In this study an encryption method is present by which each character occupies less than a byte and will save huge mobile set memory space and at the same time present encryption method will save server memory space.

Key words: Encryption, SMS, memory space

INTRODUCTION

Short Message Service (SMS) is a globally accepted wireless service that enables the transmission of alphanumeric message between mobile subscribers and external systems such as electronic mail, paging and voice-mail systems. SMS was originally designed as part of the Global System for Mobile Communications (GSM) digital mobile phone standard, but is now available on a wide range of networks, including 3G networks. The service makes use of a Short Message service Center (SMSC), which acts as a store and forward system for short message. The wireless network provides the mechanisms required to find the destination station(s) and transport short messages between the SMSCs and wireless stations. By using SMS an active mobile handset is able to receive or submit a short message at any time, independent of whether a voice or data call is in progress. SMS also guarantees delivery of the short message by the network.

Traditionally one character occupies one byte or eight bits. So if we can represent one character by only five bits, it will save huge mobile set memory space. As SMS is an offline message service and if receiving stations are not identified, the short message is stored in the SMSC until the destination device becomes available. If the large numbers of short message is stored in the SMSC, obviously it is huge pressure for SMSC. So if we can reduce each character space that it occupies, SMSC will be relieved from huge pressure.

Process: Traditionally in SMS we use the character set { a, b, c, d, ……., z}, {A, B, C, D,……, Z} , and some special characters such as , , . , ?,□, : and etc. Normally in SMS we write our text either by using lowercase letters or by using uppercase letters. Usually subscribers do not wants to change the case (upper to lower or lower to upper) in SMS. For example, in SMS we normally write congratulations. how are you? or CONGRATULATIONS, HOW ARE YOU?

In present method we exploit this characteristic of SMS. We categorize the total character set that is commonly used in SMS into two groups. Each group contains 32 characters. Group 1 contains all 26 lowercase letters and 6 most common special characters and the Group 2 contains all 26 uppercase letters and 6 commonly used special characters. That is:

Group 1

a = 00000	r = 10001
b = 00001	s = 10010
c = 00010	t = 10011
d = 00011	u = 10100
e = 00100	v = 10101
f = 00101	w = 10110
g = 00110	x = 10111
h = 00111	y = 11000
i = 01000	z = 11001
j = 01001	, = 11010
k = 01010	. = 11011
l = 01011	□ = 11100
m = 01100	: = 11110
n = 01101	& = 11110
o = 01110	
p = 01111	
q = 10000	

Corresponding Author: Mohammad Abu Yousuf, Department of Computer Science and Engineering, Mawlana Bhashani Science and Technology University, Santosh, Tangail-1902, Bangladesh

Group 2

A = 00000	R = 10001
B = 00001	S = 10010
C = 00010	T = 10011
D = 00011	U = 10100
E = 00100	V = 10101
F = 00101	W = 10110
G = 00110	X = 10111
H = 00111	Y = 11000
I = 01000	Z = 11001
J = 01001	, = 11010
K = 01010	. = 11011
L = 01011	□ = 11100
M = 01100	: = 11110
N = 01101	& = 11110
O = 01110	
P = 01111	
Q = 10000	

Fig.1: Binary representation of character sets of Group1 and Group2

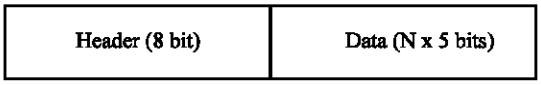

Fig.2: Frame structure

Group 1: { a, b, c, d, ……..,z, , , . , ?,□, : and} = 32 characters
Group 2: {A, B, C, D,……, Z, , , . , ?,□, : and } = 32 characters

As in each group the total numbers of characters are 32, so we need only 5 bits to represent all characters in any group. That is $2^5 = 32$. Now let us consider the binary representation of all characters in each group. Binary representations of all characters are illustrated in Fig.1.

First, consider the 160 byte as 160X8 a bit stream. If four or more consecutive characters (Suppose N characters) come from the same group present encryption method will reduce the required bits from Nx8 bits to (Nx5+8) bits to represent the information. The consecutive characters of same group will be treated as a frame, which has two parts. First part is known as Header and the second part is known as Data. Header part contains 8 bits and Data part contains Nx5 bits, where N is the number of consecutive characters. Frame structure is illustrated in Fig.2.

In Header first 3 most significant bits (MSB), bit position 5, 6, 7 are header identification bits. Next 1 bit, bit position 4 is used for group identification and last 4 bits, bit position 0, 1, 2, 3 represent the total numbers of consecutive characters from same group. Header structure is illustrated in Fig.3.

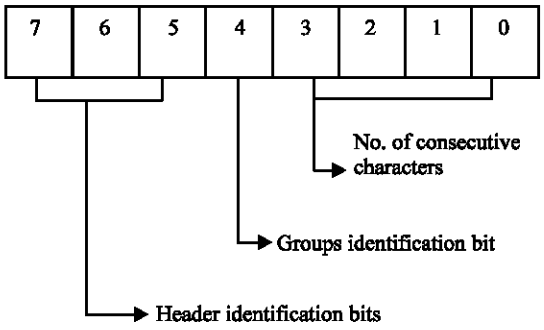

Fig.3: Header structure

- If the first 3 most significant bits (MSB), bit position 5, 6, 7 of any byte are 000, it will be identified as header. Otherwise it will be treat as uncompressed character. From the ASCII character set we find any visible character must have 1 in any position in first 3 most significant bits (MSB).
- Next 1 bit, bit position 4 represents group no. If bit is 1, it represents Group 1 and if 0 it represents Group 2.
- Bit position 0 to 3, 4 bits represents number of consecutive characters from same group, coming next. For example if bits are 0111, the numbers of consecutive characters from same group are 7, so next 7 characters will be in compressed mode.

Using 4 bits we can represent maximum value of 15. So in our method we need a header for each 15 compressed characters. If we have 27 consecutive characters from the same group, we have to use 2 headers. 1 header will identify first 15 characters then second header will identify next 12 characters.

ALGORITHM

The algorithm of the present encryption method is transcribed in Fig.4.

IMPLEMENTATION

Here is a simple example that demonstrates how the present encryption method saves memory spaces.

A simple text is given below:
congratulation. how are you?'

There are 28 characters in the above-mentioned text. At first 'congratulation.', consecutive 15 characters of group1, so it will take 1 header 8bit + 5X15=83 bits and rest 13 characters '□ how_are_you?' will take another 1 header 8bit + 5X13=73 bits. In total (83+ 73) = 156 bits means 19.5 bytes = 20 bytes instead of 28 bytes in normal SMS .

```
Define NON_Group 2
Define Group1  1
Define Group2  0

InputBuffer      [Stores sms]
OutputBuffer     [Stores Compressed sms]

SameGroupCounter = 0
RunningGroup = NON_Group
I = 0
While not end of data
      X = InputBuffer[I]
      G = FindGroup(X)
      If RunningGroup  = = G then
          SameGroupCounter = SameGroupCounter + 1
          If SameGroupCounter ==15 and RunningGroup<2 then
                Add header to OutputBuffer  [1 byte/8 bit]
                Add Last SameGroupCounter characters from I of InputBuffer to OutputBuffer
in compressed form [5 bit each]
          End if
      Else
          If SameGroupCounter >= 5 and RunningGroup < 2  then
                Add header to OutputBuffer  [1 byte/8 bit]
                Add Last SameGroupCounter characters from I of InputBuffer to OutputBuffer
in compressed form [5 bit each]
          Else
               Add Last SameGroupCounter Characters from I of InputBuffer to OutputBuffer
in ASCII form [8 bit each]
          End if
          SameGroupCounter = 1
          RunningGroup = G
      End if
      I = I + 1
End While
```

Fig. 4: The encryption algorithm

So by using present encryption method to send the above text it is required:

In the above text all characters from the same group, Group1.

In the above text at first 15 consecutive characters from Group 1 'congratulation.' comes
So in the frame the Header part is like

- First 3 bits (MSB) are 000, which represents it is header.
- Next bit is 1, which represents the character from Group 1.
- Next 4 bits are 1111, which represents numbers of consecutive characters from same group (group 1) are 15. That is 'congratulation.'

Data part contains 5X15 = 75 bits because numbers of consecutive characters from Group1 are 15 and each character occupies 5 bits of memory space. Frame structure for character sets 'congratulation.' is illustrated in Fig.5.

So total number of bits required for 'congratulation.' are (8+5X15) = 83 bits. Here 8 bits for Header part.
After characters 'congratulation.' next 13 consecutive characters from Group 1 '_how_are_you?' comes

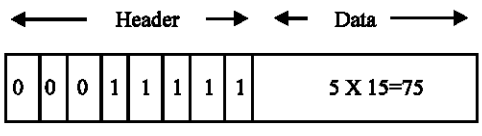

Fig.5: Frame structure for character sets 'congratulation.'

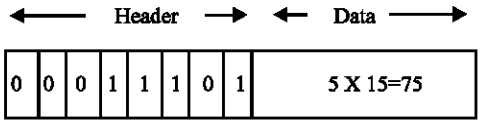

Fig.6: Frame structure for character sets '_how_are_you?'

So in the frame the Header part is like

- First 3 bits (MSB) are 000, which represents it is header.
- Next bit is 1, which represents the character from Group 1.
- Next 4 bits are 1101, which represents numbers of consecutive characters from same group (group 1) are 13. That is '_how_are_you?'.

Data part contains 5X13 = 65 bits because numbers of consecutive characters from Group1 are 13 and each character occupies 5 bits of memory space. Frame structure for character sets '□ how □ are□ you?' is illustrated in Fig.6.

So total number of bits required for 'how are you?' are (8+5X13) = 73 bits. Here 8 bits for Header part.

So the total number of bits required for the whole text 'congratulation. how are you?' are:

8 + 5X15 = 83 bits
8 + 5X13 = 73 bits

Total 156 bits

156 bits means 19.5 bytes = 20 bytes. So it is possible to save (28 –20) = 8 bytes of memory space to sent the above-mentioned 28 characters length text.

PERFORMANCE ANALYSIS OF THE PRESENT METHOD

Best case: If 15 consecutive characters come from same group, it will be the best case. For example, the text is 'congratulation.' Here total numbers of characters are 15 from the same group, Group1.

Normally one character occupies 8 bits or one byte of memory space. So traditionally to send the above text in SMS it is required 15X8 = 120 bits.

By using present encryption method to send the above text it is required (8 + 15X5) = 83 bits. Here 8 bits for Header of the frame and 15X5 bits for Data. Each character occupies 5 bits and total numbers of characters are 15.

So it is possible to save maximum (120 – 83) = 37 bits of memory space. That is, it is possible to save 30.83% of memory space for the best case.

Worst case: 'ConGraTulATion' where there is no consecutive 4 or more characters from same group. No compression will take place. So space will not reduce.

CONCLUSIONS

This study presents a new encryption method, by which we can reduce the required memory space. Thus it will reduce the communication pressure on mobile network as well as will reduce the required memory space in server. From the customers' characteristics we know that they don't want to change the case (upper or lower) because to change letter case it needs to press extra several keys. For typing SMS customers use only thumb (1 finger). So, it's obvious that without special purpose users will not change the case and will use the letter of the same group. So this encryption method will increase overall performance of mobile messaging system.

REFERENCES

1. Scott, B., M. Guthery, J. Cronin, 2002. Mobile Application Development with SMS and the SIM Toolkit, McGraw-Hill Companies,
2. Gwenael, L.B., 2003. Mobile Messaging Technologies and Services: SMS, EMS and MMS, Wiley Computer Publishing, John Wiley and Sons, Inc.,
3. Amaud, H.L., J. Vincent, 2004. SMS and MMS Interworking in Mobile Networks, Artech House, Inc.
4. Donald, J.L., 2003. Wireless Messaging Demystified: SMS, EMS, MMS, IM and others, McGraw-Hill Companies.
5. Martyn, M., 2003. Mobile and Wireless Design Essentials, Wiley Publishing, John Wiley and Sons, Inc.
6. Yi, B.L., Imrich Chlamtac, 2001. Wireless and Mobile Network Architechture, Wiley Computer Publishing, John Wiley and Sons, Inc.