IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CID 1800 Comment Resolution | | | | |
| Date: 2015-05-13 | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Paul Lambert | Marvell Inc |  |  |  |
|  |  |  |  |  |

Abstract

This document provides proposed comment resolutions for follow comments:

CID #1800

|  |  |
| --- | --- |
| 1800 | Hash function require crc32 calculation per hash, should be much more efficient |

##### 10.25.3.4.4 Bloom filter Hash function operation

The Service Hint Map field is formed by XORing together the bloom id for each service to be represented in the field. The bloom ids and bloom filter are determined by:

*k* - the Number of Hash Functions carried in the Bloom Filter Information field.

M - is the length in octets of the bloom filter (Service Hint field length -2)

Ui – the USID of the i-th service to be mapped into a Bloom Id

A Bloom Id is formed by setting k bit positions in in the Service Hint Map using k hash functions.

for i in range(k):

set bit H(i,Ui ,M)

The hash functionH(i,U,M) is computed as follows:

Let U[ i : i+2 ] represent 2 sequential octets in the octet string U

Step 1: Compute A(i,U) = U[ i mod 8, (i mod 8)+2] XOR U[ (int(i/8)+8, int(i/8)+10)

Step 2: H(i,U,M) = A(i,U) mod *M\*8*

The hash function H(i,U,M) supports up Bloom Ids with Number of Hash Functions (k) up to 54.

The first 16 Bloom Id hash functions H(i,U,M) are shown in Table 10-16a.

Table 10-16a – Hash functions for the Bloom Id

|  |  |
| --- | --- |
| **Hash Function**  **Index *i*** | **Hash Function** |
| 0 | U[0:2] XOR U[8:10] mod M\*8 |
| 1 | U[1:3] XOR U[8:10] mod M\*8 |
| 2 | U[2:4] XOR U[8:10] mod M\*8 |
| 3 | U[3:5] XOR U[8:10] mod M\*8 |
| 4 | U[4:6] XOR U[8:10] mod M\*8 |
| 5 | U[5:7] XOR U[8:10] mod M\*8 |
| 6 | U[6:8] XOR U[8:10] mod M\*8 |
| 7 | U[7:9] XOR U[8:10] mod M\*8 |
| 8 | U[0:2] XOR U[9:11] mod M\*8 |
| 9 | U[1:3] XOR U[9:11] mod M\*8 |
| 10 | U[2:4] XOR U[9:11] mod M\*8 |
| 11 | U[3:5] XOR U[9:11] mod M\*8 |
| 12 | U[4:6] XOR U[9:11] mod M\*8 |
| 13 | U[5:7] XOR U[9:11] mod M\*8 |
| 14 | U[6:8] XOR U[9:11] mod M\*8 |
| 15 | U[7:9] XOR U[9:11] mod M\*8 |

The Number of Hash Functions field is used to indicate the number of hash functions, *k*, (out of the maximum of 16) used by the Bloom filter. For example, 0001 means the first 2 hash functions are used (denoted by hash function index 0x00 and 0x01, as shown in Table 10-16a).

### ZA.4.1: Determining the Bloom Filter Size, *m*

*Reference: Annex A* [B56]

A Bloom filter is a space-efficient probabilistic data structure used to test if an element (e.g., service) is a member of a set. A Bloom filter is an array of *m* bits, representing a set of *n* services S={x1, x2, …, x*n*}. These *m*-bits are initially set to all zero. A service x, is mapped to a random number uniformly between 1, …, *m* by using *k* hash functions, *hi*(*k*), for 1≤ *i* ≤ *k*.

A service y is reported as a member of S, if the bits *hi*(*y*) are set to all ones, and is guaranteed to not to be a member of S if any bit, *hi*(*y*), is set to zero.

*p* is the probability of the false positive event (lower bound)*,* which occurs when y is actually not a member of S, but reported as being in the set, is given by formula (1) and is dependent on the parameters *n*, *m* and *k.*

The variables *p*, *n*, *m* and *k* are related to each other with the following approximation, formula (1):

(1)

The optimal value of *k,* is given by formula (2):

(2)

Substituting k from (2) in (1) and reordering terms, the value of *m,* rounded to the nearest   
multiple of 8, is given by formula (3):

(3)

For example, for *n*=25 services and *p*=0.01, the size of the Bloom filter *m* is 240 bits and the required number of hash function is 7.