IEEE P802.11
Wireless LANs

|  |
| --- |
| Combining Service Hashes |
| Date: July 15, 2015 |
| Author(s): |
| Name | Company | Address | Phone | email |
| Payam Torab | Broadcom |  |  | ptorab@broadcom.com |

Abstract

This submission proposes to extend the Service Hash element (8.4.2.173) to allow a general boolean function to combine the service hashes included in the element. The current boolean function is an implicit OR (any). The submission is provided as a resolution to CID 1463.

**Revision History**

R0: Initial revision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID 1463** | **Category** | **Page**  | **Subclause** | **Line** |
| The Service Hash element is not flexible enough to allow defining an arbitrary Boolean combination of services (service hash values). The implicit relationship assumed is a Boolean OR (ANY) when seeking services, and seemingly AND (ALL) when indicating services. For example, a STA that inserts two services s1 and s2 in the Service Hash element is interpreted as being interested in s1 OR s2, prompting a response by any AP that provides s1 alone, s2 alone, or both s1 and s2.This is a common use case but does not cover all applications. A STA may be exclusively interested in APs that provide BOTH s1 and s2. For example, a mobile device may be searching for a docking station that is equipped with BOTH a mouse AND a keyboard: In this case Including "mouse" and "keyboard" services in the current form of Service Hash element will trigger responses by docking stations that provide "mouse" or "keyboard" but not both, in addition to docking stations that provide both, which in turn results in increased message exchange at base data rates, air pollution, power consumption etc.The Boolean function implicitly assumed in the Service Hash element needs to be extended from the implicit OR function (S1 + S2+ ... + SN) (when searching) and AND function (S1 . S2 . ... . SN) (when advertising) to a canonical form such as sum-of-products (SoP) or product-of-sums (PoS) to allow an arbitrary combination of services. This will result in fewer message exchanges when certain combination of services is of interest.The encoding of the canonical form can be optimized for the current (implicit OR, implicit AND) semantics. | Technical | 10 | 8.4.2.173 | 30 |

# 1. Background

The Service Hash element (8.4.2.173) is defined as a linear set of 6-octet Service Hash values (also referred to as Service Hash, Hash, Hash Value, with upper case or with lower case in this discussion), with each value representing a service of interest. When seeking services, the implicit function combining the service values in the Service Hash element is the logical OR function (meaning ANY), for example,

**10.25.3.4.3 Solicited PAD procedure**

An AP having dot11SolictedPADActivated equals to true shall include Service Advertisement element (8.4.2.172) in Probe Response frame, if there is one or more Service Hashes (8.4.2.173) matching with the received Probe Request containing the Service Hash element sent by the non-AP STA.

On the advertising side, when indicating available services, the implicit function combining the subfields of the Basic Service Information Descriptors field of the Service Advertisement element (8.4.2.172) is the logical AND function (meaning ALL).

To summarize, a STA that includes two services S1 and S2 in the Service Hash element is interpreted as being interested in S1 OR S2, prompting a response by any service provider that provides S1 alone, S2 alone, or both S1 and S2.

There is reason to extend both of these implicit functions,

* When seeking a service, a STA may be exclusively interested in service providers that provide a combination of services. For example,
	+ In dense environments a mobile device searching for an AP that provides a set of services will trigger responses by many APs that provide at least one of those services. Applications such as Wi-Fi offload will lead to seeking a more sophisticated combination of services and could lead to response floods.



<https://meraki.cisco.com/solutions/high-density-wifi>

* + A mobile device searching for a docking station that is equipped with BOTH a mouse AND a keyboard. In this case Including “mouse” and “keyboard” services in the current form of Service Hash element will trigger responses by docking stations that provide “mouse” or “keyboard” but not both, in addition to docking stations that provide both “mouse” and “keyboard”. Another example is a STA searching for an AP that provides LOCATION and Time OF DAY services. The absence of semantics to specify interest in all services and not any of them results in increased message exchange at base data rates, air pollution, power consumption etc.
* When advertising a service, a service provider may have resource or policy conflicts to provide a certain combination of services. For example, a docking station may be able to provide “storage” or “charging” services, but not both at the same time. There is currently no way to formulate such restrictions.

It is possible to extend the implicit functions assumed in these scenarios from the implicit OR function (S1 + S2 + ... + SN) (when searching) and implicit AND function (S1 . S2 . ... . SN) (when advertising) to more flexible functions. The rest of the discussion is about encoding of such functions.

# 2. Encoding (representation)

The most general representation of a boolean function of N boolean variables is the canonical sum-of-products (SOP) (or product-of-sums (POS)) form. Expressing an arbitrary boolean function of N boolean variables as a sum of up to 2N minterms with SOP (or product of 2N maxterms with POS) requires 2N bits or ⎡2N/8⎤ octets.

We observe however, that a subset of boolean functions are not of interest in the service discovery context. Specifically, it is not meaningful for a seeker to include a service S just to indicate that it is not seeking it, and similarly, it is not meaningful for provider to include a service S just t indicate that it is not providing it.

What this means is boolean functions of interest for N services S1, S2, ..., SN can be written as a sum-of-positive-products

F(S1, S2, ..., SN) = S1 . (product of some of the advertised services other than S1) +

 S2 . (product of some of the advertised services other than S2) +

 ...

 SN . (product of some of the advertised services other than SN) (1)

Or by an NxN *combination* matrix

$$A=\left[\begin{matrix}1&a\_{12}&…&a\_{1N}\\a\_{21}&1&…&a\_{2N}\\\vdots &\vdots &\ddots &\vdots \\a\_{N1}&a\_{N2}&…&1\end{matrix}\right]$$

where aij = 1 if the term i in the sum above includes Sj, i,j = 1, 2, ..., N.

For example, the boolean function F(S1, S2, S3, S4) = S1S2 + S2S3 + S3S1 + S4S1S2 can be represented by the 4x4 combination matrix

$$\left[\begin{matrix}1&1&0&0\\0&1&1&0\\1&0&1&0\\1&1&0&1\end{matrix}\right]$$

Since all diagonal elements of a combination matrix are 1, it suffices to represent it by its N(N-1) off-diagonal elements. Thus, boolean functions of interest for service discovery can be represented by ⎡N(N-1)/8⎤ octets instead of ⎡2N/8⎤ octets. The following table demonstrates the saving in number of bytes.

|  |  |  |  |
| --- | --- | --- | --- |
| Number of ServicesN | Sum-of-products form(⎣2N/8⎦ octets) | Compressed form(⎣N(N-1)/8⎦ octets) | Saving in number of octets |
| 1 | 0 | 0 | 0% |
| 2 | 1 | 1 | 0% |
| 3 | 1 | 1 | 0% |
| 4 | 2 | 2 | 0% |
| 5 | 4 | 3 | 25% |
| 6 | 8 | 3 | 62.5% |
| 7 | 16 | 6 | 62.5% |
| 8 | 32 | 7 | 78.125% |

Finally, two special cases of interest are signaled independently to save more bytes,

* ANY of the N listed services (OR), combination matrix equal to identity matrix (**I**N×N)
* ALL of the N listed services (AND), all combination matrix elements equal to 1 (**1**N×N)

# 3. Processing (receiver rules)

Service advertisers receiving the Service Hash element are required to process the all the elements.

*[Note for editor: Modify the following paragraph].*

**8.4.2.173 Service Hash element**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Element ID | Length | Service Hash Value | Flags | Combination |
| Octets | 1 | 1 | 6*N*, where *N* is the number of services present in the element  | 1 | ⎡*N*(*N*-1)/8⎤ octets where *N* is thenumber of services present in the element |

The Flags field is defined in Figure 8-xxx (Flags field format).

|  |  |  |  |
| --- | --- | --- | --- |
|  | B0 | B1 | B2 B7 |
|  | Any | All | Reserved |
| Bits: | 1 | 1 | 6 |

**Figure 8-xxx—Flags field format**