|  |  |
| --- | --- |
| Project | **IEEE 802.21 MIHS****<**[**http://www.ieee802.org/21/**](http://www.ieee802.org/21/)**>** |
| Title | **An Algorithm for Complete Subtree Creation** |
| DCN | **21-13-0205-00-MuGM** |
| Date Submitted | **November 11, 2013** |
| Source(s) | Yoshihiro Ohba (Toshiba) |  |
| Re: | IEEE 802.21 Session #59 in Dallas |
| Abstract | This document describes an algorithm for generating complete subtree for GKB. |
| Purpose | To addresses LB7a Comment #136. |
| Notice | This document has been prepared to assist the IEEE 802.21 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. |
| Release | The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that IEEE 802.21 may make this contribution public. |
| Patent Policy | The contributor is familiar with IEEE patent policy, as stated in [Section 6 of the IEEE-SA Standards Board bylaws](http://standards.ieee.org/guides/opman/sect6.html#6.3) <[http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://127.0.0.1:4664/cache?event_id=757737&schema_id=1&s=5X0vID10lu_E6yrIkWkNd4Wz2H8&q=hancock)> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/faq.pdf> |

# **Proposal**

Add the following python code to 9.4.2.5.1.

def CreateCompleteSubtree(I, T, R):

 # Input I: List of indices of leaf nodes to be included in the group

 # Input T: The entire tree that covers all leaf nodes

 # Input R: Root node of the entire tree

 # Output S: Complete Subtree for the group.

 S=[]

 def check(n):

 # Input n: subtree root node

 # Output 0, 1

 # 0 : Some node in the subtree is a non-member of the group.

 # 1 : All nodes in the subtree are members of the group.

 if n.left==None and n.right==None: # n is leaf

 if n.index.val in I:

 S.append(n)

 return 1

 return 0

 # n is non-leaf

 lval=check(n.left)

 rval=check(n.right)

 if lval\*rval>0:

 S.remove(n.left)

 S.remove(n.right)

 S.append(n)

 return 1

 return 0

 check(R)

 return S