Analysis of OLT3540 & OLCDV auto-reboot because of CFM stack overflow during Y1731 test

作者 : 余祖波(livelylittlefish@gmail.com)
Blog: http://blog.csdn.net/livelylittlefish, http://www.abo321.org

Content

0. register and instruction basis

LR: Link Register, to save the return value of calling function
CR: Condition Register, to save the result of an algorithm, and provides conditional-branch. CR is consist of 8 4-bits fields, crf0 ~ crf7 respectively.
CTR: Counter Register, for loop, and will decrease according the specific branching operation
GPR: General Purpose Register, and there are 32 GRPs, r0 ~ r31.

r1: used to save the stack pointer
r3 ~ r10: parameters passed to callee, r3 is the first parameter, r4 is the second, and so on.
r3 is also used to save the return value.

mfspr rD, SPR ;read SPR (Special Purpose Register) to rD, 即读取特殊功能寄存器
mtspr SPR, rS ;write rS to SPR (Special Purpose Register), 即写入特殊功能寄存器

lbz rD, d(rA) ;move (rA)+d from memory (the lowest 8 bits, that is, 1 byte) to rD, 即高位清零加载字节指令（偏移地址寻址）, byte
lhz rD, d(rA) ;move (rA)+d from memory (the lowest 16 bits, that is, 2 bytes) to rD, 即高位清零的加载半字指令（偏移地址寻址）, half word
lwz rD, d(rA) ;move (rA)+d from memory (the total 32 bits, that is, 4 bytes) to rD, 即加载字指令（偏移地址寻址）, word

stw rS, d(rA) ;move (rS) (32 bits) to (rA)+d
;即字存储指令 stw（偏移地址寻址）,将 rS 的 32 位内容存储到有效地址为(rA)+d 的存储器中

stwu rS, d(rA) ;move (rS) (32 bits) to (rA)+d, and (rA)=(rA)+d
;即记录有效地址的字存储指令 stw（偏移地址寻址）,将 rS 的 32 位内容存储到有效地址为(rA)+d 的存储器中

li rD, value ;load value to rD, 将 value 加载到寄存器 rD

1. split the call stack, and map the stack frames to functions

(1) backtrace to find all functions in the call path

According to GRP1, the general register, which saved the stack pointer, We backtrace to find all functions in the call path. Below it is shown in green background.

GPR (1) - 0x4f5e3c0 stack pointer, respond to %esp in ia32 architecture

---

```
3540> dumpexc 5

******   Exception Record   ******

Time       = MON AUG 30 04:53:40 2010
Exception  = DSI Exception - 0x300
Task Name  = tCfmAg
Saved stack pointer    = 0x04f5f0c8
```

```
Bottom of the stack     = 0x04f5ec38
Actual end of stack     = 0x04f5cc38
Stack Size              = 0x00002000

0x04f5e3c0

Print Task 0x4f5ec38 stask starts:

        0x04f5ec38 - 0x00000000        0x04f5ec34 - 0x00000000        0x04f5ec30 - 0x00000000        0x04f5ec2c - 0x00000000
        0x04f5ec28 - 0x00000000        0x04f5ec24 - 0x00000000        0x04f5ec20 - 0x00000000        0x04f5ec1c - 0x00000000
        0x04f5ec18 - 0x00000000        0x04f5ec14 - 0x00000000        0x04f5ec10 - 0x00000000        0x04f5ec0c - 0xeeeeeeee
        0x04f5ec08 - 0xeeeeeeee        0x04f5ec04 - 0xeeeeeeee        0x04f5ec00 - 0xeeeeeeee        0x04f5ebfc - 0xeeeeeeee
        0x04f5ebf8 - 0xeeeeeeee        0x04f5ebf4 - 0x00000000

        (2.9) vxTaskEntry stack frame

        0x04f5ebf0 - 0x00000000        0x04f5ebec - 0xeeeeeeee
        0x04f5ebe8 - 0xeeeeeeee        0x04f5ebe4 - 0xeeeeeeee
        0x04f5ebe0 - 0xeeeeeeee        0x04f5ebdc - 0x00000000
        0x04f5ebd8 - 0x00000000
        0x04f5ebd4 - 0x00541548  //return address of calling cfm_agOlcEntry

        (2.8) cfm_agOlcEntry stack frame

        0x04f5ebd0 - 0x04f5ebf0        0x04f5ebcc - 0x00000000
        0x04f5ebc8 - 0x0021a6f8        0x04f5ebc4 - 0x00000000        0x04f5ebc0 - 0x00000000        0x04f5ebbc - 0x00000000
        0x04f5ebb8 - 0xeeeeeeee        0x04f5ebb4 - 0xeeeeeeee        0x04f5ebb0 - 0xeeeeeeee        0x04f5ebac - 0xeeeeeeee
        0x04f5eba8 - 0xeeeeeeee        0x04f5eba4 - 0xeeeeeeee        0x04f5eba0 - 0x00df9e58        0x04f5eb9c - 0xeeeeeeee
        0x04f5eb98 - 0x09810840
        0x04f5eb94 - 0x0021a82c  //return address of calling cfm_agSysWideTimerExpire

        (2.7) cfm_agSysWideTimerExpire stack frame
```

0x04f5eb90 - 0x04f5ebd0
0x04f5eb8c - 0x00cd0e96
0x04f5eb88 - 0x0000004c
0x04f5eb84 - 0x006d0000
0x04f5eb80 - 0x00000000
0x04f5eb7c - 0x00000064
0x04f5eb78 - 0x00df9e58
0x04f5eb74 - 0x00228dc8 //return address of calling cfm_agSMCCI_timerUpdate

(2.6) cfm_agSMCCI_timerUpdate stack frame

0x04f5eb70 - 0x04f5eb90
0x04f5eb6c - 0xffffffff
0x04f5eb68 - 0x0000001f
0x04f5eb64 - 0x00222844 //return address of calling cfm_agSMCCI_dispatchEvent

(2.5) cfm_agSMCCI_dispatchEvent stack frame

The following stack frame is of the function void cfm_agSMCCI_dispatchEvent(dot1agMP_t *agMP, agCCIEvents_t event), there is no any local variables in this function, then, the space of its stack frame is only 12 bytes.
The same as (2.2)

0x04f5eb60 - 0x04f5eb70
0x04f5eb5c - 0x00000064
0x04f5eb58 - 0x00000000
0x04f5eb54 - 0x0022869c //return address of calling cfm_agSMCCI_dispatchEvent

(2.4) cfm_agSMCCI_waiting stack frame

The following stack

void cfm_agSMCCI_waiting(dot1agMP_t *agMP, agCCIEvents_t event)

```
{
    //...

    if(!agMP->mp.mep.CCIenabled)
    {
        cfm_agSMCCI_enterState(agMP, CCI_STATE_IDLE);
        return;
    }
}
```

0x04f5eb50 - 0x04f5eb60
0x04f5eb4c - 0x00cd0e96
0x04f5eb48 - 0x04fc6e60
0x04f5eb44 - 0x006423c8
0x04f5eb40 - 0x04f5eb70
0x04f5eb3c - 0x03609450
0x04f5eb38 - 0x04f5eb78
0x04f5eb34 - 0x00222a98 //return address of calling cfm_agSMCCI_enterState(agMP, CCI_STATE_IDLE);

(2.3) cfm_agSMCCI_enterState stack frame

The following stack frame is the following function. There is no any local variables in this function, then, the space of its stack frame is only 12 bytes.

```
void cfm_agSMCCI_enterState(dot1agMP_t *agMP, agCCIStates_t newState)
{
    if(agMP==NULL || newState>=CCI_STATE_LAST)
        return;

    agMP->mp.mep.CCIstate=newState;
    cfm_agSMCCI_dispatchEvent(agMP, CCI_EVENT_ENTER_STATE);
}
```

0x04f5eb30 - 0x04f5eb50
0x04f5eb2c - 0xffffffff
0x04f5eb28 - 0x00000000
0x04f5eb24 - 0x002286e0 //return value of calling cfm_agSMCCI_dispatchEvent

(2.2) cfm_agSMCCI_dispatchEvent stack frame

The following stack frame is of the following function, there is no any local variables in this function, then, the space of its stack frame is only 16 bytes.

```
void cfm_agSMCCI_dispatchEvent(dot1agMP_t *agMP, agCCIEvents_t event)
{
/*
 * The CCI_EVENT_ENTER_STATE event should only be called by
 * agSMCCI_enterState(), and NOT directly.
 */
   if(agMP==NULL || event>=CCI_EVENT_LAST)
       return;

   switch(event)
   {
   case CCI_EVENT_ENTER_STATE:
       break;

   case CCI_EVENT_TIME_OUT:
       agMP->mp.mep.CCIwhile=0;
       break;

   case CCI_EVENT_ENABLE_CHANGE:
   case CCI_EVENT_MAC_CHANGE:
       break;
```

```
    default:
        break;
    }

    agSMCCIStateFuncArray[agMP->mp.mep.CCIstate](agMP,event);
}
```

0x04f5eb20 - 0x04f5eb30
0x04f5eb1c - 0x00000000
0x04f5eb18 - 0x00000000
0x04f5eb14 - 0x0022869c //return address of calling cfm_agSMCCI_waiting

(2.1) cfm_agSMCCI_waiting stack frame

```
void cfm_agSMCCI_waiting(dot1agMP_t *agMP, agCCIEvents_t event)
{
    //...
    switch(event)
    {
    case CCI_EVENT_ENTER_STATE:
        cfm_agXmitCCM(agMP);

        //agMP->mp.mep.CCIwhile=(agMP->association->CCMInterval*1000);
        agMP->mp.mep.CCIwhile=(agMP->association->CCMInterval);    //the address of this instruction is 0x00222a04
        cfm_agSMCCI_startTimer(agMP, agMP->mp.mep.CCIwhile);

        //...
    }

    //...
}
```

0x04f5eb10 - 0x04f5eb20
0x04f5eb0c - 0x04fc6e60
0x04f5eb08 - 0x00000001
0x04f5eb04 - 0x0098e2a4
0x04f5eb00 - 0x0000b032
0x04f5eafc - 0x00cd0eb0
0x04f5eaf8 - 0x00000000
0x04f5eaf4 - 0x00222a04 //return address of calling cfm_agXmitCCM

(2) cfm_agXmitCCM stack frame

this following stack space is 0x04f5eaf0 – 0x04f5e4f0 = 0x600, that is, 1536 bytes. Then, according this and the disassemble code shown below, we can make sure that the following stack frame is of the following function. Its detailed analysis is recorded in another file.

```
void cfm_agXmitCCM(dot1agMP_t *agMP)
{
    uint32_t  bytesLeft;
    uint8_t   buffer[DOT1AG_MAX_MTU_SIZE];   // #define DOT1AG_MAX_MTU_SIZE  1500
    //...

    cfm_agSendNPPktMulticast(agMP, buffer, sizeof(buffer)-bytesLeft, FALSE, DOT1AG_PDU_CCM_PKT);
}
```

```
OLCDV> l cfm_agXmitCCM
                        cfm_agXmitCCM:
0x143eec  9421fa00    stwu        r1,-1536(r1)    //allocate 1536 bytes in the stack
0x143ef0  7c0802a6    mfspr       r0,LR
0x143ef4  93a105f4    stw         r29,1524(r1)
0x143ef8  93e105fc    stw         r31,1532(r1)
0x143efc  90010604    stw         r0,1540(r1)
0x143f00  7c7f1b78    or          r31,r3,r3
```

```
0x143f04  88a3001e    lbz         r5,30(r3)
0x143f08  81630010    lwz         r11,16(r3)
0x143f0c  81230010    lwz         r9,16(r3)
0x143f10  81290018    lwz         r9,24(r9)
0x143f14  88e90004    lbz         r7,4(r9)
0x143f18  a1030026    lhz         r8,38(r3)
0x143f1c  3c600053    lis         r3,0x53 # 83
0x143f20  38633eb8    addi        r3,r3,0x3eb8 # 16056
0x143f24  809f0014    lwz         r4,20(r31)
0x143f28  80cb001c    lwz         r6,28(r11)
...
0x144060  20bd05dc    subfic      r5,r29,0x5dc # 1500  //the third parameter, sizeof(buffer)-bytesLeft
0x144064  7fe3fb78    or          r3,r31,r31           //the first parameter, agMP
0x144068  38810010    addi        r4,r1,0x10 # 16
0x14406c  38c00000    li          r6,0x0 # 0
0x144070  38e00005    li          r7,0x5 # 5
0x144074  4800fa89    bl          0x153afc # cfm_agSendNPPktMulticast
...
```

```
                                                  0x04f5eaf0 - 0x04f5eb10       0x04f5eaec - 0x04fc6e60
      0x04f5eae8 - 0x01b60000       0x04f5eae4 - 0x04fc6de0       0x04f5eae0 - 0x04fb67b8       0x04f5eadc - 0x01c92a6c
      0x04f5ead8 - 0x00000000       0x04f5ead4 - 0x00000000       0x04f5ead0 - 0x00000000       0x04f5eacc - 0x00000000
      0x04f5eac8 - 0x00000000       0x04f5eac4 - 0x00000000       0x04f5eac0 - 0x00000000       0x04f5eabc - 0x00000000
      0x04f5eab8 - 0x00000000       0x04f5eab4 - 0x00000000       0x04f5eab0 - 0x00000000       0x04f5eaac - 0x00000000
      0x04f5eaa8 - 0x00000000       0x04f5eaa4 - 0x00000000       0x04f5eaa0 - 0x00000000       0x04f5ea9c - 0x00000000
      0x04f5ea98 - 0x00000000       0x04f5ea94 - 0x00000000       0x04f5ea90 - 0x00000000       0x04f5ea8c - 0x00000000
      0x04f5ea88 - 0x00000000       0x04f5ea84 - 0x00000000       0x04f5ea80 - 0x00000000       0x04f5ea7c - 0x00000000
      0x04f5ea78 - 0x00000000       0x04f5ea74 - 0x00000000       0x04f5ea70 - 0x00000000       0x04f5ea6c - 0x00000000
      0x04f5ea68 - 0x00000000       0x04f5ea64 - 0x00000000       0x04f5ea60 - 0x00000000       0x04f5ea5c - 0x00000000
      0x04f5ea58 - 0x00000000       0x04f5ea54 - 0x00000000       0x04f5ea50 - 0x00000000       0x04f5ea4c - 0x00000000
      0x04f5ea48 - 0x00000000       0x04f5ea44 - 0x00000000       0x04f5ea40 - 0x00000000       0x04f5ea3c - 0x00000000
      0x04f5ea38 - 0x00000000       0x04f5ea34 - 0x00000000       0x04f5ea30 - 0x00000000       0x04f5ea2c - 0x00000000
      0x04f5ea28 - 0x00000000       0x04f5ea24 - 0x00000000       0x04f5ea20 - 0x00000000       0x04f5ea1c - 0x00000000
```

0x04f5ea18 - 0x00000000	0x04f5ea14 - 0x00000000	0x04f5ea10 - 0x00000000	0x04f5ea0c - 0x00000000
0x04f5ea08 - 0x00000000	0x04f5ea04 - 0x00000000	0x04f5ea00 - 0x00000000	0x04f5e9fc - 0x00000000
0x04f5e9f8 - 0x00000000	0x04f5e9f4 - 0x00000000	0x04f5e9f0 - 0x00000000	0x04f5e9ec - 0x00000000
0x04f5e9e8 - 0x00000000	0x04f5e9e4 - 0x00000000	0x04f5e9e0 - 0x00000000	0x04f5e9dc - 0x00000000
0x04f5e9d8 - 0x00000000	0x04f5e9d4 - 0x00000000	0x04f5e9d0 - 0x00000000	0x04f5e9cc - 0x00000000
0x04f5e9c8 - 0x00000000	0x04f5e9c4 - 0x00000000	0x04f5e9c0 - 0x00000000	0x04f5e9bc - 0x00000000
0x04f5e9b8 - 0x00000000	0x04f5e9b4 - 0x00000000	0x04f5e9b0 - 0x00000000	0x04f5e9ac - 0x00000000
0x04f5e9a8 - 0x00000000	0x04f5e9a4 - 0x00000000	0x04f5e9a0 - 0x00000000	0x04f5e99c - 0x00000000
0x04f5e998 - 0x00000000	0x04f5e994 - 0x00000000	0x04f5e990 - 0x00000000	0x04f5e98c - 0x00000000
0x04f5e988 - 0x00000000	0x04f5e984 - 0x00000000	0x04f5e980 - 0x00000000	0x04f5e97c - 0x00000000
0x04f5e978 - 0x00000000	0x04f5e974 - 0x00000000	0x04f5e970 - 0x00000000	0x04f5e96c - 0x00000000
0x04f5e968 - 0x00000000	0x04f5e964 - 0x00000000	0x04f5e960 - 0x00000000	0x04f5e95c - 0x00000000
0x04f5e958 - 0x00000000	0x04f5e954 - 0x00000000	0x04f5e950 - 0x00000000	0x04f5e94c - 0x00000000
0x04f5e948 - 0x00000000	0x04f5e944 - 0x00000000	0x04f5e940 - 0x00000000	0x04f5e93c - 0x00000000
0x04f5e938 - 0x00000000	0x04f5e934 - 0x00000000	0x04f5e930 - 0x00000000	0x04f5e92c - 0x00000000
0x04f5e928 - 0x00000000	0x04f5e924 - 0x00000000	0x04f5e920 - 0x00000000	0x04f5e91c - 0x00000000
0x04f5e918 - 0x00000000	0x04f5e914 - 0x00000000	0x04f5e910 - 0x00000000	0x04f5e90c - 0x00000000
0x04f5e908 - 0x00000000	0x04f5e904 - 0x00000000	0x04f5e900 - 0x00000000	0x04f5e8fc - 0x00000000
0x04f5e8f8 - 0x00000000	0x04f5e8f4 - 0x00000000	0x04f5e8f0 - 0x00000000	0x04f5e8ec - 0x00000000
0x04f5e8e8 - 0x00000000	0x04f5e8e4 - 0x00000000	0x04f5e8e0 - 0x00000000	0x04f5e8dc - 0x00000000
0x04f5e8d8 - 0x00000000	0x04f5e8d4 - 0x00000000	0x04f5e8d0 - 0x00000000	0x04f5e8cc - 0x00000000
0x04f5e8c8 - 0x00000000	0x04f5e8c4 - 0x00000000	0x04f5e8c0 - 0x00000000	0x04f5e8bc - 0x00000000
0x04f5e8b8 - 0x00000000	0x04f5e8b4 - 0x00000000	0x04f5e8b0 - 0x00000000	0x04f5e8ac - 0x00000000
0x04f5e8a8 - 0x00000000	0x04f5e8a4 - 0x00000000	0x04f5e8a0 - 0x00000000	0x04f5e89c - 0x00000000
0x04f5e898 - 0x00000000	0x04f5e894 - 0x00000000	0x04f5e890 - 0x00000000	0x04f5e88c - 0x00000000
0x04f5e888 - 0x00000000	0x04f5e884 - 0x00000000	0x04f5e880 - 0x00000000	0x04f5e87c - 0x00000000
0x04f5e878 - 0x00000000	0x04f5e874 - 0x00000000	0x04f5e870 - 0x00000000	0x04f5e86c - 0x00000000
0x04f5e868 - 0x00000000	0x04f5e864 - 0x00000000	0x04f5e860 - 0x00000000	0x04f5e85c - 0x00000000
0x04f5e858 - 0x00000000	0x04f5e854 - 0x00000000	0x04f5e850 - 0x00000000	0x04f5e84c - 0x00000000
0x04f5e848 - 0x00000000	0x04f5e844 - 0x00000000	0x04f5e840 - 0x00000000	0x04f5e83c - 0x00000000
0x04f5e838 - 0x00000000	0x04f5e834 - 0x00000000	0x04f5e830 - 0x00000000	0x04f5e82c - 0x00000000
0x04f5e828 - 0x00000000	0x04f5e824 - 0x00000000	0x04f5e820 - 0x00000000	0x04f5e81c - 0x00000000
0x04f5e818 - 0x00000000	0x04f5e814 - 0x00000000	0x04f5e810 - 0x00000000	0x04f5e80c - 0x00000000

0x04f5e808 - 0x00000000	0x04f5e804 - 0x00000000	0x04f5e800 - 0x00000000	0x04f5e7fc - 0x00000000
0x04f5e7f8 - 0x00000000	0x04f5e7f4 - 0x00000000	0x04f5e7f0 - 0x00000000	0x04f5e7ec - 0x00000000
0x04f5e7e8 - 0x00000000	0x04f5e7e4 - 0x00000000	0x04f5e7e0 - 0x00000000	0x04f5e7dc - 0x00000000
0x04f5e7d8 - 0x00000000	0x04f5e7d4 - 0x00000000	0x04f5e7d0 - 0x00000000	0x04f5e7cc - 0x00000000
0x04f5e7c8 - 0x00000000	0x04f5e7c4 - 0x00000000	0x04f5e7c0 - 0x00000000	0x04f5e7bc - 0x00000000
0x04f5e7b8 - 0x00000000	0x04f5e7b4 - 0x00000000	0x04f5e7b0 - 0x00000000	0x04f5e7ac - 0x00000000
0x04f5e7a8 - 0x00000000	0x04f5e7a4 - 0x00000000	0x04f5e7a0 - 0x00000000	0x04f5e79c - 0x00000000
0x04f5e798 - 0x00000000	0x04f5e794 - 0x00000000	0x04f5e790 - 0x00000000	0x04f5e78c - 0x00000000
0x04f5e788 - 0x00000000	0x04f5e784 - 0x00000000	0x04f5e780 - 0x00000000	0x04f5e77c - 0x00000000
0x04f5e778 - 0x00000000	0x04f5e774 - 0x00000000	0x04f5e770 - 0x00000000	0x04f5e76c - 0x00000000
0x04f5e768 - 0x00000000	0x04f5e764 - 0x00000000	0x04f5e760 - 0x00000000	0x04f5e75c - 0x00000000
0x04f5e758 - 0x00000000	0x04f5e754 - 0x00000000	0x04f5e750 - 0x00000000	0x04f5e74c - 0x00000000
0x04f5e748 - 0x00000000	0x04f5e744 - 0x00000000	0x04f5e740 - 0x00000000	0x04f5e73c - 0x00000000
0x04f5e738 - 0x00000000	0x04f5e734 - 0x00000000	0x04f5e730 - 0x00000000	0x04f5e72c - 0x00000000
0x04f5e728 - 0x00000000	0x04f5e724 - 0x00000000	0x04f5e720 - 0x00000000	0x04f5e71c - 0x00000000
0x04f5e718 - 0x00000000	0x04f5e714 - 0x00000000	0x04f5e710 - 0x00000000	0x04f5e70c - 0x00000000
0x04f5e708 - 0x00000000	0x04f5e704 - 0x00000000	0x04f5e700 - 0x00000000	0x04f5e6fc - 0x00000000
0x04f5e6f8 - 0x00000000	0x04f5e6f4 - 0x00000000	0x04f5e6f0 - 0x00000000	0x04f5e6ec - 0x00000000
0x04f5e6e8 - 0x00000000	0x04f5e6e4 - 0x00000000	0x04f5e6e0 - 0x00000000	0x04f5e6dc - 0x00000000
0x04f5e6d8 - 0x00000000	0x04f5e6d4 - 0x00000000	0x04f5e6d0 - 0x00000000	0x04f5e6cc - 0x00000000
0x04f5e6c8 - 0x00000000	0x04f5e6c4 - 0x00000000	0x04f5e6c0 - 0x00000000	0x04f5e6bc - 0x00000000
0x04f5e6b8 - 0x00000000	0x04f5e6b4 - 0x00000000	0x04f5e6b0 - 0x00000000	0x04f5e6ac - 0x00000000
0x04f5e6a8 - 0x00000000	0x04f5e6a4 - 0x00000000	0x04f5e6a0 - 0x00000000	0x04f5e69c - 0x00000000
0x04f5e698 - 0x00000000	0x04f5e694 - 0x00000000	0x04f5e690 - 0x00000000	0x04f5e68c - 0x00000000
0x04f5e688 - 0x00000000	0x04f5e684 - 0x00000000	0x04f5e680 - 0x00000000	0x04f5e67c - 0x00000000
0x04f5e678 - 0x00000000	0x04f5e674 - 0x00000000	0x04f5e670 - 0x00000000	0x04f5e66c - 0x00000000
0x04f5e668 - 0x00000000	0x04f5e664 - 0x00000000	0x04f5e660 - 0x00000000	0x04f5e65c - 0x00000000
0x04f5e658 - 0x00000000	0x04f5e654 - 0x00000000	0x04f5e650 - 0x00000000	0x04f5e64c - 0x00000000
0x04f5e648 - 0x00000000	0x04f5e644 - 0x00000000	0x04f5e640 - 0x00000000	0x04f5e63c - 0x00000000
0x04f5e638 - 0x00000000	0x04f5e634 - 0x00000000	0x04f5e630 - 0x00000000	0x04f5e62c - 0x00000000
0x04f5e628 - 0x00000000	0x04f5e624 - 0x00000000	0x04f5e620 - 0x00000000	0x04f5e61c - 0x00000000
0x04f5e618 - 0x00000000	0x04f5e614 - 0x00000000	0x04f5e610 - 0x00000000	0x04f5e60c - 0x00000000
0x04f5e608 - 0x00000000	0x04f5e604 - 0x00000000	0x04f5e600 - 0x00000000	0x04f5e5fc - 0x00000000

```
0x04f5e5f8 - 0x00000000        0x04f5e5f4 - 0x00000000        0x04f5e5f0 - 0x00000000        0x04f5e5ec - 0x00000000
0x04f5e5e8 - 0x00000000        0x04f5e5e4 - 0x00000000        0x04f5e5e0 - 0x00000000        0x04f5e5dc - 0x00000000
0x04f5e5d8 - 0x00000000        0x04f5e5d4 - 0x00000000        0x04f5e5d0 - 0x00000000        0x04f5e5cc - 0x00000000
0x04f5e5c8 - 0x00000000        0x04f5e5c4 - 0x00000000        0x04f5e5c0 - 0x00000000        0x04f5e5bc - 0x00000000
0x04f5e5b8 - 0x00000000        0x04f5e5b4 - 0x00000000        0x04f5e5b0 - 0x00000000        0x04f5e5ac - 0x00000000
0x04f5e5a8 - 0x00000000        0x04f5e5a4 - 0x00000000        0x04f5e5a0 - 0x00000000        0x04f5e59c - 0x00000000
0x04f5e598 - 0x00000000        0x04f5e594 - 0x00000000        0x04f5e590 - 0x00000000        0x04f5e58c - 0x00000000
0x04f5e588 - 0x00000000        0x04f5e584 - 0x00000000        0x04f5e580 - 0x00000000        0x04f5e57c - 0x00000000
0x04f5e578 - 0x00000000        0x04f5e574 - 0x01020000        0x04f5e570 - 0x01020400        0x04f5e56c - 0x00a10200
0x04f5e568 - 0x0a682889        0x04f5e564 - 0x06010106        0x04f5e560 - 0x062b0601        0x04f5e55c - 0x01000f00
0x04f5e558 - 0x00000000        0x04f5e554 - 0x00000000        0x04f5e550 - 0x00000000        0x04f5e54c - 0x00000000
0x04f5e548 - 0x00000000        0x04f5e544 - 0x00000000        0x04f5e540 - 0x00000000        0x04f5e53c - 0x00000000
0x04f5e538 - 0x00000000        0x04f5e534 - 0x00000000        0x04f5e530 - 0x6c697261        0x04f5e52c - 0x02067361
0x04f5e528 - 0x656c2034        0x04f5e524 - 0x206c6576        0x04f5e520 - 0x31373331        0x04f5e51c - 0x040e592e
0x04f5e518 - 0x1a550064        0x04f5e514 - 0x85460000        0x04f5e510 - 0x89028001        0x04f5e50c - 0x81000064
0x04f5e508 - 0x49011765
0x04f5e504 - 0x00340005
0x04f5e500 - 0x0180c200
0x04f5e4fc - 0x00000000
0x04f5e4f8 - 0x00000000
0x04f5e4f4 - 0x00221d64  //return address of calling cfm_agSendNPPKtMulticast
```

(2.10) cfm_agSendNPPKtMulticast stack frame

```
void cfm_agSendNPPktMulticast(dot1agMP_t *agMP, uint8_t *pdu, uint32_t pduSize, uint8_t forceOut, uint8_t type)

0x04f5e4f0 - 0x04f5eaf0        0x04f5e4ec - 0x04fc6e60
0x04f5e4e8 - 0x00000000        0x04f5e4e4 - 0x00000565
0x04f5e4e0 - 0x04fb67b8        0x04f5e4dc - 0x00000064
0x04f5e4d8 - 0x00000000        0x04f5e4d4 - 0x00000000
0x04f5e4d0 - 0x20000084        0x04f5e4cc - 0x00000000
0x04f5e4c8 - 0x00000000        0x04f5e4c4 - 0x002318d0
```

(2.11) cfm_agIMEPEgressProcessing stack frame

```
void cfm_agIMEPEgressProcessing(dot1agMP_t *agMP, uint8_t *pdu, uint32_t pduSize)

0x04f5e4c0 - 0x04f5e4f0        0x04f5e4bc - 0x04fc7c70
0x04f5e4b8 - 0x04fc6e60        0x04f5e4b4 - 0x00000000        0x04f5e4b0 - 0x04f5e500        0x04f5e4ac - 0x04fc6e60
0x04f5e4a8 - 0x00000010        0x04f5e4a4 - 0x1766056e        0x04f5e4a0 - 0x00054901        0x04f5e49c - 0x04f5e500
0x04f5e498 - 0x01000000        0x04f5e494 - 0x0021fda8
```

(2.12) cfm_agCCReceiverBlk stack frame

```
void cfm_agCCReceiverBlk(dot1agMP_t *agMP, uint8_t *pdu, uint32_t pduSize, uint8_t forceOut)
```

```
OLCDV> l cfm_agCCReceiverBlk,200
                      cfm_agCCReceiverBlk:
0x14409c  9421ff70    stwu        r1,-144(r1)
0x1440a0  7c0802a6    mfspr       r0,LR
0x1440a4  92c10068    stw         r22,104(r1)
0x1440a8  92e1006c    stw         r23,108(r1)
0x1440ac  93010070    stw         r24,112(r1)
0x1440b0  93210074    stw         r25,116(r1)
0x1440b4  93410078    stw         r26,120(r1)
0x1440b8  9361007c    stw         r27,124(r1)
0x1440bc  93810080    stw         r28,128(r1)
0x1440c0  93a10084    stw         r29,132(r1)
0x1440c4  93c10088    stw         r30,136(r1)
0x1440c8  93e1008c    stw         r31,140(r1)
0x1440cc  90010094    stw         r0,148(r1)
0x1440d0  7c7f1b78    or          r31,r3,r3
0x1440d4  7c9c2378    or          r28,r4,r4
0x1440d8  7cbb2b78    or          r27,r5,r5
0x1440dc  7cd63378    or          r22,r6,r6
...
```

```
0x14422c  807f0010    lwz         r3,16(r31)
0x144230  7ee4bb78    or          r4,r23,r23
0x144234  4bffbb89    bl          0x13fdbc # cfm_agFindCliRmepByID
...
```

```
        0x04f5e490 - 0x04f5e4c0        0x04f5e48c - 0x04f5e512
        0x04f5e488 - 0x04fc7c70        0x04f5e484 - 0x04f5e500        0x04f5e480 - 0x00000077        0x04f5e47c - 0x00000077
        0x04f5e478 - 0x00000005        0x04f5e474 - 0x00000000        0x04f5e470 - 0x00000000        0x04f5e46c - 0x00000000
        0x04f5e468 - 0x00000000        0x04f5e464 - 0x00000000        0x04f5e460 - 0x04f5e480        0x04f5e45c - 0x00000000
        0x04f5e458 - 0x00000000        0x04f5e454 - 0x00000000        0x04f5e450 - 0x00000000        0x04f5e44c - 0x00000000
        0x04f5e448 - 0x00000083        0x04f5e444 - 0x0037f240        0x04f5e440 - 0x04f5e480        0x04f5e43c - 0x00000000
        0x04f5e438 - 0x0000002c        0x04f5e434 - 0x08800000        0x04f5e430 - 0x00001d55        0x04f5e42c - 0x04f5e488
        0x04f5e428 - 0x00000000        0x04f5e424 - 0x0038af78        0x04f5e420 - 0x04f5e440        0x04f5e41c - 0x00000000
        0x04f5e418 - 0x0000002c
        0x04f5e414 - 0x000000b0
        0x04f5e410 - 0x04f5e440
        0x04f5e40c - 0x0000002c
        0x04f5e408 - 0x08800000
        0x04f5e404 - 0x00221f24  //(r1)+36, the return address of calling cfm_agFindCliRmepByID
```

(3) stack frame of cfm_agFindCliRmepByID

```
        0x04f5e400 - 0x04f5e490
        0x04f5e3fc - 0x04fc7c70  //r31 => (r1)+28
        0x04f5e3f8 - 0x04f5e512  //r30 => (r1)+24
        0x04f5e3f4 - 0x04f5e500
        0x04f5e3f0 - 0x04f5e500
        0x04f5e3ec - 0x000013a2
        0x04f5e3e8 - 0x00000000
        0x04f5e3e4 - 0x0021daf0  //(r1)+4   //return address of calling cfm_agGetNextCliRmep
```

```
dot1agCliRmep_t *cfm_agFindCliRmepByID(dot1agAssociation_t *association, uint16_t mepid)
{
```

```
    dot1agCliRmep_t *rmep=NULL;

    if(association==NULL)
        return(NULL);

    while((rmep=cfm_agGetNextCliRmep(association, rmep))!=NULL)
    {
        if(rmep->mepid==mepid)
            return(rmep);
    }
    return(NULL);
}

OLCDV> l cfm_agFindCliRmepByID, 28
                        cfm_agFindCliRmepByID:

//save registers and allocate space
0x13fdbc  9421ffe0    stwu        r1,-32(r1)    /*(r1) = (r1) - 32, (r1) is 0x04f5e400, after this instruction, it will be 0x04f5e3e0,
                                                  Then, 0x04f5e3e0 will be the next callee's stack frame base pointer, and the callee is
                                                  cfm_agGetNextCliRmep
                                                 */
0x13fdc0  7c0802a6    mfspr       r0,LR         //move LR (Link Register, save return address of calling function) to r0, (LR)=0x00221f24

0x13fdc4  93c10018    stw         r30,24(r1)    //move (r30) to (r1)+24, that is, (r30)=0x04f5e512
0x13fdc8  93e1001c    stw         r31,28(r1)    //move (r31) to (r1)+28, that is, (r31)=0x04fc7c70
0x13fdcc  90010024    stw         r0,36(r1)     //move (r0) to (r1)+36, at this time, (r0)=0x00221f24, then, (r1)+36 content is 0x00221f24

//save parameters
0x13fdd0  7c7f1b78    or          r31,r3,r3     //(r3) => r31, r3 is the first parameter, that is, 'dot1agAssociation_t *association'
0x13fdd4  7c9e2378    or          r30,r4,r4     //(r4) => r30, r4 is the second parameter, that is, 'uint16_t mepid'

0x13fdd8  38800000    li          r4,0x0 # 0    //0 => r4, r4 is the local variable, rmep
0x13fddc  38600000    li          r3,0x0 # 0    //0 => r3
```

```
    if(association==NULL)
        return(NULL);
0x13fde0  2f9f0000    cmpi        crf7,0,r31,0x0 # 0            //r31 is the first parameter, association
0x13fde4  419e0030    bc          0xc,30, 0x13fe14 # 0x0013fe14  //return NULL; (r3)=NULL is the return value

0x13fde8  48000014    b           0x13fdfc # 0x0013fdfc

    while((rmep=cfm_agGetNextCliRmep(association, rmep))!=NULL)
    {
        if(rmep->mepid==mepid)
            return(rmep);
    }

//the following is the while-loop
0x13fdec  a0040016    lhz         r0,22(r4)           //(r4)+22 => r0, and the lowest 16 bits, that is, rmep->mepid => r0
0x13fdf0  7c832378    or          r3,r4,r4            //(r4) => r3, r4 is the local variable, rmep
0x13fdf4  7f80f000    cmp         crf7,0,r0,r30
0x13fdf8  419e001c    bc          0xc,30, 0x13fe14 # 0x0013fe14  //return rmep; (r3)=rmep is the return value

0x13fdfc  7fe3fb78    or          r3,r31,r31          //(r31) => r3, that is, the first parameter 'dot1agAssociation_t *association'
0x13fe00  4bfff981    bl          0x13f780 # cfm_agGetNextCliRmep
                                              /* rmep=fm_agGetNextCliRmep(association, rmep)
                                                 r3 is association, a pointer, the first parameter;
                                                 r4 is the local variable, rmep, and its initialization value is NULL, or
                                                 r4 is the return value rmep (by moveing r3 to r4 below)
                                               */

0x13fe04  7c641b78    or          r4,r3,r3            //the return value, rmep is in r3, then, copy it to r4
0x13fe08  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x13fe0c  409effe0    bc          0x4,30, 0x13fdec # 0x0013fdec

    return(NULL);
```

```
0x13fe10  38600000    li          r3,0x0 # 0          //if not found, move NULL to r3 as the return value

//restore registers and stack pointer
0x13fe14  80010024    lwz         r0,36(r1)           //(r1)+36 => r0, (r1)+36 is the return address
0x13fe18  7c0803a6    mtspr       LR,r0               //r0 => LR, the return address is saved into LR

0x13fe1c  83c10018    lwz         r30,24(r1)          //(r1)+24 => r30
0x13fe20  83e1001c    lwz         r31,28(r1)          //(r1)+28 => r31
0x13fe24  38210020    addi        r1,r1,0x20 # 32     //(r1)+32 => r1, that is, release the allocated stack space
0x13fe28  4e800020    blr                             //return
value = 0 = 0x0
```

(4) cfm_agGetNextCliRmep stack frame

```
        0x04f5e3e0 - 0x04f5e400
        0x04f5e3dc - 0x04168570
        0x04f5e3d8 - 0x0000002c
        0x04f5e3d4 - 0x04161918  //the garbage data
```

```
dot1agCliRmep_t *cfm_agGetNextCliRmep(dot1agAssociation_t *association, dot1agCliRmep_t *rmep)
{
    p2btree *p2bNode;

    if(association==NULL)
        return(NULL);

    if(rmep==NULL)
        p2bNode=p2BtreeFirst(&association->cliRmepList);
    else
        p2bNode=p2BtreeNext(&rmep->cliRmepNode);    //this instruction does be called in the call path, then, rmep!=NULL

    if(p2bNode==NULL)
        return(NULL);
```

```
    return(STRUCT(dot1agCliRmep_t, cliRmepNode, p2bNode));   //this instruction is optimized by the compiler
}


3540> l cfm_agGetNextCliRmep,23
                        cfm_agGetNextCliRmep:
0x21d46c  9421fff0    stwu        r1,-16(r1)
0x21d470  7c0802a6    mfspr       r0,LR
0x21d474  90010014    stw         r0,20(r1)

0x21d478  7c691b78    or          r9,r3,r3   //(r3) => r9, save the first parameter association to r9
0x21d47c  7c832378    or          r3,r4,r4   //(r4) => r3, save the second parameter rmep to r3

0x21d480  38000000    li          r0,0x0 # 0 //0 => r0

    if(association==NULL)
        return(NULL);
0x21d484  2f890000    cmpi        crf7,0,r9,0x0 # 0     //if(association==NULL)
0x21d488  419e002c    bc          0xc,30, 0x21d4b4 # 0x0021d4b4

    if(rmep==NULL)
        p2bNode=p2BtreeFirst(&association->cliRmepList);
    else
        p2bNode=p2BtreeNext(&rmep->cliRmepNode);
0x21d48c  2f840000    cmpi        crf7,0,r4,0x0 # 0     //if(rmep==NULL)
0x21d490  40be0010    bc          0x5,30, 0x21d4a0 # 0x0021d4a0

0x21d494  38690014    addi        r3,r9,0x14 # 20       //(r9)+20 => r3, that is, &association->cliRmepList
0x21d498  4bffbfa5    bl          0x21943c # p2BtreeFirst
0x21d49c  48000008    b           0x21d4a4 # 0x0021d4a4

0x21d4a0  4bffbdd5    bl          0x219274 # p2BtreeNext
```

```
    if(p2bNode==NULL)
        return(NULL);
0x21d4a4  38000000    li          r0,0x0 # 0            /*0 => r0, here, 0x21d4a4 is the return address of calling p2BtreeNext.
                                                          and ,(LR)=0x21d4a4, and in the future of this call path, it will not change.
                                                        */
0x21d4a8  2f830000    cmpi        crf7,0,r3,0x0 # 0     //r3 is the return value, p2bNode,
0x21d4ac  419e0008    bc          0xc,30, 0x21d4b4 # 0x0021d4b4

0x21d4b0  7c601b78    or          r0,r3,r3              //r3 => r0, is p2bNode, the return value

0x21d4b4  7c030378    or          r3,r0,r0              //(r0) => r3, that is, NULL, as the return value
0x21d4b8  80010014    lwz         r0,20(r1)
0x21d4bc  7c0803a6    mtspr       LR,r0
0x21d4c0  38210010    addi        r1,r1,0x10 # 16
0x21d4c4  4e800020    blr
value = 0 = 0x0
3540>
```

(5) p2BtreeNext stack frame

```
        0x04f5e3d0 - 0x04f5e3e0
        0x04f5e3cc - 0x04f5e418
        0x04f5e3c8 - 0x00000000
        0x04f5e3c4 - 0x00000000

//default alignment is 4 bytes.
typedef struct p2BalTree {
    struct p2BalTree    *b_bwd;
    struct p2BalTree    *b_left;
    struct p2BalTree    *b_right;
    signed char         b_bal;
} p2btree,*p2btree_pt;
```

```
2btree *p2BtreeNext(p2btree * nd)
{
    /* if there is right link, follow it then all left links. */
    if (nd->b_right) {
        nd = nd->b_right;
        for (; nd; nd = nd->b_left) {
            if (!nd->b_left)
                return (nd);
        }
    }
    /* go up tree till we find a link that is left (ie. up is greater) */
    while (1) {
        if (nd->b_bwd == 0)
            return (0);
        if (nd->b_bwd->b_left == nd)
            return (nd->b_bwd);
        nd = nd->b_bwd;
    }
    return (0);
}
```

```
           ---------------
0x41be0016 |             |  +0, b_bwd
           ---------------
           |             |  +4, b_left
           ---------------
           |             |  +8, b_right
           ---------------
           |             |  +12, b_bal
           ---------------
           |             |
           ---------------
```

```
3540> l p2BtreeNext, 29
                        p2BtreeNext:
0x219274  9421fff0    stwu        r1,-16(r1)    //(r1)-16 => r1, then, r1=0x04f5e3c0, the stack pointer, and from here, we backtrace
0x219278  80030008    lwz         r0,8(r3)      /*(r3) is the first parameter, p2btree* nd, (r3)+8 => r0, (r0)=0x41be0024=nd->b_right
                                                  Then, (r3)=0x41be0016, that is, nd=0x41be0016
                                                 */


    if (nd->b_right) {
0x21927c  2f800000    cmpi        crf7,0,r0,0x0 # 0
0x219280  419e002c    bc          0xc,30, 0x2192ac # 0x002192ac

        nd = nd->b_right;
0x219284  80630008    lwz         r3,8(r3)      //(r3)+8 => r3, (r3)+8 is nd->b_right, then, new nd' is nd->b_right, saved in r3
0x219288  2f830000    cmpi        crf7,0,r3,0x0 # 0   //'nd;' in for-loop, that is, nd!=0
0x21928c  419e0020    bc          0xc,30, 0x2192ac # 0x002192ac

        for (; nd; nd = nd->b_left) {
            if (!nd->b_left)
                return (nd);
        }
0x219290  80030004    lwz         r0,4(r3)      //(r3)+4 => r0, (r3)+4 is nd'->b_left
0x219294  7c691b78    or          r9,r3,r3      //(r3) => r9, nd' saved in r9
0x219298  2f800000    cmpi        crf7,0,r0,0x0 # 0   // if (!nd->b_left)
0x21929c  419e0040    bc          0xc,30, 0x2192dc # 0x002192dc

0x2192a0  80630004    lwz         r3,4(r3)      //(r3+4) => r3, (r3)+4 is nd->b_left
0x2192a4  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x2192a8  409effe8    bc          0x4,30, 0x219290 # 0x00219290

    while (1) {
        if (nd->b_bwd == 0)
            return (0);
        if (nd->b_bwd->b_left == nd)
```

```
            return (nd->b_bwd);
        nd = nd->b_bwd;
    }

0x2192ac  80030000    lwz         r0,0(r3)      //(r3)+0 => r0, (r3)+0 is nd->b_bwd, saved in r0
0x2192b0  39200000    li          r9,0x0 # 0    //0 => r9
0x2192b4  2f800000    cmpi        crf7,0,r0,0x0 # 0
0x2192b8  419e0024    bc          0xc,30, 0x2192dc # 0x002192dc

0x2192bc  81230000    lwz         r9,0(r3)      //nd->b_bwd => r9
0x2192c0  80090004    lwz         r0,4(r9)      //nd->b_bwd->b_left => r0
0x2192c4  7f801800    cmp         crf7,0,r0,r3
0x2192c8  40be000c    bc          0x5,30, 0x2192d4 # 0x002192d4

0x2192cc  81230000    lwz         r9,0(r3)                //nd->b_bwd => r9
0x2192d0  4800000c    b           0x2192dc # 0x002192dc   //jump 0x2192dc, then, return nd->b_bwd;

0x2192d4  80630000    lwz         r3,0(r3)                //nd->b_bwd => r3
0x2192d8  4bffffd4    b           0x2192ac # 0x002192ac   //continue loop

    return (0);
0x2192dc  7d234b78    or          r3,r9,r9                //(r9)=0 is the return value
0x2192e0  38210010    addi        r1,r1,0x10 # 16
0x2192e4  4e800020    blr
value = 0 = 0x0
3540>

        0x04f5e3c0 - 0x04f5e3d0       0x04f5e3bc - 0x00000000
        0x04f5e3b8 - 0x08800000       0x04f5e3b4 - 0x0039652c       0x04f5e3b0 - 0x04f5e3e0       0x04f5e3ac - 0x0000002c
        0x04f5e3a8 - 0x00000000       0x04f5e3a4 - 0x00392d64       0x04f5e3a0 - 0x04f5e3e0       0x04f5e39c - 0x00a84b08
        0x04f5e398 - 0x0000002c       0x04f5e394 - 0x00000008       0x04f5e390 - 0x04fb67b8       0x04f5e38c - 0x000000c8
        0x04f5e388 - 0x00000004       0x04f5e384 - 0x00000028       0x04f5e380 - 0x00000001       0x04f5e37c - 0x09000000
        0x04f5e378 - 0x0073770c       0x04f5e374 - 0x0039652c       0x04f5e370 - 0x04f5e400       0x04f5e36c - 0x0000002c
```

0x04f5e368 - 0x00000000	0x04f5e364 - 0x00000000	0x04f5e360 - 0x04f5e420	0x04f5e35c - 0xffffffff
0x04f5e358 - 0x00000000	0x04f5e354 - 0x00000001	0x04f5e350 - 0x04f5e3e8	0x04f5e34c - 0x04f5ec38
0x04f5e348 - 0x08800000	0x04f5e344 - 0x00396070	0x04f5e340 - 0x04f5e370	0x04f5e33c - 0x08800000
0x04f5e338 - 0x0082c944	0x04f5e334 - 0x00000001	0x04f5e330 - 0x04f5e3f0	0x04f5e32c - 0x80280064
0x04f5e328 - 0x0083025c	0x04f5e324 - 0x00000000	0x04f5e320 - 0x04f5e3e0	0x04f5e31c - 0x80280064
0x04f5e318 - 0x0083025c	0x04f5e314 - 0x00000000	0x04f5e310 - 0x0082ac64	0x04f5e30c - 0x08800000
0x04f5e308 - 0x0082c944	0x04f5e304 - 0x00000001	0x04f5e300 - 0x0082ac64	0x04f5e2fc - 0x80280064
0x04f5e2f8 - 0x0083025c	0x04f5e2f4 - 0x00000000	0x04f5e2f0 - 0x04f5e3b0	0x04f5e2ec - 0x00000000
0x04f5e2e8 - 0x00000000	0x04f5e2e4 - 0x00000000	0x04f5e2e0 - 0x0505282b	0x04f5e2dc - 0x00000003
0x04f5e2d8 - 0x08800000	0x04f5e2d4 - 0x00000000	0x04f5e2d0 - 0x0082ac64	0x04f5e2cc - 0x08800000
0x04f5e2c8 - 0x0082c944	0x04f5e2c4 - 0x00000001	0x04f5e2c0 - 0x04f5e380	0x04f5e2bc - 0x80280064
0x04f5e2b8 - 0x0083025c	0x04f5e2b4 - 0x00000000	0x04f5e2b0 - 0x04f5e370	0x04f5e2ac - 0x00000000
0x04f5e2a8 - 0x00000000	0x04f5e2a4 - 0x00000000	0x04f5e2a0 - 0x0505282f	0x04f5e29c - 0x00000003
0x04f5e298 - 0x08800000	0x04f5e294 - 0x00000000	0x04f5e290 - 0x00000000	0x04f5e28c - 0x00000000
0x04f5e288 - 0x00000000	0x04f5e284 - 0x00000000	0x04f5e280 - 0x04f5e340	0x04f5e27c - 0x00000000
0x04f5e278 - 0x00000000	0x04f5e274 - 0x00000000	0x04f5e270 - 0x00000000	0x04f5e26c - 0x00000000
0x04f5e268 - 0x00000000	0x04f5e264 - 0x00000000	0x04f5e260 - 0x00000000	0x04f5e25c - 0x00000000
0x04f5e258 - 0x00000000	0x04f5e254 - 0x00000000	0x04f5e250 - 0x00000000	0x04f5e24c - 0x00000000
0x04f5e248 - 0x00000000	0x04f5e244 - 0x00000000	0x04f5e240 - 0x00000000	0x04f5e23c - 0x00000000
0x04f5e238 - 0x00000000	0x04f5e234 - 0x00000000	0x04f5e230 - 0x00000000	0x04f5e22c - 0x00000000
0x04f5e228 - 0x00000000	0x04f5e224 - 0x00000000	0x04f5e220 - 0x00000000	0x04f5e21c - 0x00000000
0x04f5e218 - 0x00000000	0x04f5e214 - 0x00000000	0x04f5e210 - 0x00000000	0x04f5e20c - 0x00000000
0x04f5e208 - 0x00000000	0x04f5e204 - 0x00000000	0x04f5e200 - 0x00000000	0x04f5e1fc - 0x00000000
0x04f5e1f8 - 0x00000000	0x04f5e1f4 - 0x00000000	0x04f5e1f0 - 0x00000000	0x04f5e1ec - 0x00000000
0x04f5e1e8 - 0x00000000	0x04f5e1e4 - 0x00000000	0x04f5e1e0 - 0x00000000	0x04f5e1dc - 0x00000000
0x04f5e1d8 - 0x00000000	0x04f5e1d4 - 0x00000000	0x04f5e1d0 - 0x00000000	0x04f5e1cc - 0x00000000
0x04f5e1c8 - 0x00000000	0x04f5e1c4 - 0x00000000	0x04f5e1c0 - 0x00000000	0x04f5e1bc - 0x00000000
0x04f5e1b8 - 0x00000000	0x04f5e1b4 - 0x00000000	0x04f5e1b0 - 0x00000000	0x04f5e1ac - 0x00000000
0x04f5e1a8 - 0x00000000	0x04f5e1a4 - 0x00000000	0x04f5e1a0 - 0x00000000	0x04f5e19c - 0x00000000
0x04f5e198 - 0x00000000	0x04f5e194 - 0x00000000	0x04f5e190 - 0x00000000	0x04f5e18c - 0x00000000
0x04f5e188 - 0x00000000	0x04f5e184 - 0x00000000	0x04f5e180 - 0x00000000	0x04f5e17c - 0x00000000
0x04f5e178 - 0x00000000	0x04f5e174 - 0x00000000	0x04f5e170 - 0x00000000	0x04f5e16c - 0x00000000
0x04f5e168 - 0x00000000	0x04f5e164 - 0x00000000	0x04f5e160 - 0x00000000	0x04f5e15c - 0x00000000

0x04f5e158 - 0x00000000	0x04f5e154 - 0x00000000	0x04f5e150 - 0x00000000	0x04f5e14c - 0x00000000
0x04f5e148 - 0x00000000	0x04f5e144 - 0x00000000	0x04f5e140 - 0x00000000	0x04f5e13c - 0x00000000
0x04f5e138 - 0x00000000	0x04f5e134 - 0x00000000	0x04f5e130 - 0x00000000	0x04f5e12c - 0x00000000
0x04f5e128 - 0x00000000	0x04f5e124 - 0x00000000	0x04f5e120 - 0x00000000	0x04f5e11c - 0x00000000
0x04f5e118 - 0x00000000	0x04f5e114 - 0x00000000	0x04f5e110 - 0x00000000	0x04f5e10c - 0x00000000
0x04f5e108 - 0x00000000	0x04f5e104 - 0x00000000	0x04f5e100 - 0x00000000	0x04f5e0fc - 0x00000000
0x04f5e0f8 - 0x00000000	0x04f5e0f4 - 0x00000000	0x04f5e0f0 - 0x00000000	0x04f5e0ec - 0x00000000
0x04f5e0e8 - 0x00000000	0x04f5e0e4 - 0x00000000	0x04f5e0e0 - 0x00000000	0x04f5e0dc - 0x00000000
0x04f5e0d8 - 0x00000000	0x04f5e0d4 - 0x00000000	0x04f5e0d0 - 0x00000000	0x04f5e0cc - 0x00000000
0x04f5e0c8 - 0x00000000	0x04f5e0c4 - 0x00000000	0x04f5e0c0 - 0x00000000	0x04f5e0bc - 0x00000000
0x04f5e0b8 - 0x00000000	0x04f5e0b4 - 0x00000000	0x04f5e0b0 - 0x00000000	0x04f5e0ac - 0x00000000
0x04f5e0a8 - 0x00000000	0x04f5e0a4 - 0x00000000	0x04f5e0a0 - 0x00000000	0x04f5e09c - 0x00000000
0x04f5e098 - 0x00000000	0x04f5e094 - 0x00000000	0x04f5e090 - 0x00000000	0x04f5e08c - 0x00000000
0x04f5e088 - 0x00000000	0x04f5e084 - 0x00000000	0x04f5e080 - 0x00000000	0x04f5e07c - 0x00000000
0x04f5e078 - 0x00000000	0x04f5e074 - 0x00000000	0x04f5e070 - 0x00000000	0x04f5e06c - 0x00000000
0x04f5e068 - 0x00000000	0x04f5e064 - 0x00000000	0x04f5e060 - 0x00000000	0x04f5e05c - 0x00000000
0x04f5e058 - 0x00000000	0x04f5e054 - 0x00000000	0x04f5e050 - 0x00000000	0x04f5e04c - 0x00000000
0x04f5e048 - 0x00000000	0x04f5e044 - 0x00000000	0x04f5e040 - 0x00000000	0x04f5e03c - 0x00000000
0x04f5e038 - 0x00000000	0x04f5e034 - 0x00000000	0x04f5e030 - 0x00000000	0x04f5e02c - 0x00000000
0x04f5e028 - 0x00000000	0x04f5e024 - 0x00000000	0x04f5e020 - 0x00000000	0x04f5e01c - 0x00000000
0x04f5e018 - 0x00000000	0x04f5e014 - 0x00000000	0x04f5e010 - 0x00000000	0x04f5e00c - 0x00000000
0x04f5e008 - 0x00000000	0x04f5e004 - 0x00000000	0x04f5e000 - 0x00000000	0x04f5dffc - 0x00000000
0x04f5dff8 - 0x00000000	0x04f5dff4 - 0x00000000	0x04f5dff0 - 0x00000000	0x04f5dfec - 0x00000000
0x04f5dfe8 - 0x00000000	0x04f5dfe4 - 0x00000000	0x04f5dfe0 - 0x00000000	0x04f5dfdc - 0x00000000
0x04f5dfd8 - 0x00000000	0x04f5dfd4 - 0x00000000	0x04f5dfd0 - 0x00000000	0x04f5dfcc - 0x00000000
0x04f5dfc8 - 0x00000000	0x04f5dfc4 - 0x00000000	0x04f5dfc0 - 0x00000000	0x04f5dfbc - 0x00000000
0x04f5dfb8 - 0x00000000	0x04f5dfb4 - 0x00000000	0x04f5dfb0 - 0x00000000	0x04f5dfac - 0x00000000
0x04f5dfa8 - 0x00000000	0x04f5dfa4 - 0x00000000	0x04f5dfa0 - 0x00000000	0x04f5df9c - 0x00000000
0x04f5df98 - 0x00000000	0x04f5df94 - 0x00000000	0x04f5df90 - 0x00000000	0x04f5df8c - 0x00000000
0x04f5df88 - 0x00000000	0x04f5df84 - 0x00000000	0x04f5df80 - 0x00000000	0x04f5df7c - 0x00000000
0x04f5df78 - 0x00000000	0x04f5df74 - 0x00000000	0x04f5df70 - 0x00000000	0x04f5df6c - 0x00000000
0x04f5df68 - 0x00000000	0x04f5df64 - 0x00000000	0x04f5df60 - 0x00000000	0x04f5df5c - 0x00000000
0x04f5df58 - 0x00000000	0x04f5df54 - 0x00000000	0x04f5df50 - 0x00000000	0x04f5df4c - 0x00000000

0x04f5df48 - 0x00000000	0x04f5df44 - 0x00000000	0x04f5df40 - 0x00000000	0x04f5df3c - 0x00000000
0x04f5df38 - 0x00000000	0x04f5df34 - 0x00000000	0x04f5df30 - 0x00000000	0x04f5df2c - 0x00000000
0x04f5df28 - 0x00000000	0x04f5df24 - 0x00000000	0x04f5df20 - 0x00000000	0x04f5df1c - 0x00000000
0x04f5df18 - 0x00000000	0x04f5df14 - 0x2ec00000	0x04f5df10 - 0x000549e0	0x04f5df0c - 0x00000000
0x04f5df08 - 0x00000000	0x04f5df04 - 0x17650000	0x04f5df00 - 0x00054901	0x04f5defc - 0x00000000
0x04f5def8 - 0x00000000	0x04f5def4 - 0x01ac0000	0x04f5def0 - 0x00054910	0x04f5deec - 0x00000000
0x04f5dee8 - 0x00000000	0x04f5dee4 - 0x48dc0000	0x04f5dee0 - 0x000549e0	0x04f5dedc - 0x00000000
0x04f5ded8 - 0x00000000	0x04f5ded4 - 0x00225480	0x04f5ded0 - 0x04f5eb10	0x04f5decc - 0x07fb92d8
0x04f5dec8 - 0x00cd0e9e	0x04f5dec4 - 0x04f5dee0	0x04f5dec0 - 0x04f5df00	0x04f5debc - 0x00cd0eb0
0x04f5deb8 - 0x00000040	0x04f5deb4 - 0x0000003f	0x04f5deb0 - 0x07fb92d8	0x04f5deac - 0x00000000
0x04f5dea8 - 0x00000000	0x04f5dea4 - 0x00221aec	0x04f5dea0 - 0x04f5ded0	0x04f5de9c - 0x04fc6e60
0x04f5de98 - 0x07fb92d8	0x04f5de94 - 0x00cd0e9e	0x04f5de90 - 0x04f5df00	0x04f5de8c - 0x07fb92d8
0x04f5de88 - 0x09810840	0x04f5de84 - 0x1765da78	0x04f5de80 - 0x00054901	0x04f5de7c - 0x04f5e510
0x04f5de78 - 0x0afb92d8	0x04f5de74 - 0x0021fbf8	0x04f5de70 - 0x04f5dea0	0x04f5de6c - 0x00cd0eb0
0x04f5de68 - 0x04fc6e60	0x04f5de64 - 0x00cd0e9e	0x04f5de60 - 0x00000040	0x04f5de5c - 0x00cd0eb0
0x04f5de58 - 0x00000040	0x04f5de54 - 0x0000003f	0x04f5de50 - 0x00000002	0x04f5de4c - 0x09810840
0x04f5de48 - 0x00000000	0x04f5de44 - 0x00000000	0x04f5de40 - 0x00000000	0x04f5de3c - 0x00000000
0x04f5de38 - 0x00000000	0x04f5de34 - 0x00000000	0x04f5de30 - 0x00000000	0x04f5de2c - 0x00000000
0x04f5de28 - 0x00000000	0x04f5de24 - 0x00000000	0x04f5de20 - 0x00000000	0x04f5de1c - 0x00000000
0x04f5de18 - 0x00000000	0x04f5de14 - 0x00000000	0x04f5de10 - 0x00000000	0x04f5de0c - 0x00000000
0x04f5de08 - 0x00000000	0x04f5de04 - 0x00000000	0x04f5de00 - 0x00000000	0x04f5ddfc - 0x00000000
0x04f5ddf8 - 0x00000000	0x04f5ddf4 - 0x00000000	0x04f5ddf0 - 0x00000000	0x04f5ddec - 0x00000000
0x04f5dde8 - 0x00000000	0x04f5dde4 - 0x00000000	0x04f5dde0 - 0x00000000	0x04f5dddc - 0x00000000
0x04f5ddd8 - 0x00000000	0x04f5ddd4 - 0x00000000	0x04f5ddd0 - 0x00000000	0x04f5ddcc - 0x00000000
0x04f5ddc8 - 0x00000000	0x04f5ddc4 - 0x00000000	0x04f5ddc0 - 0x00000000	0x04f5ddbc - 0x00000000
0x04f5ddb8 - 0x00000000	0x04f5ddb4 - 0x00000000	0x04f5ddb0 - 0x00000000	0x04f5ddac - 0x00000000
0x04f5dda8 - 0x00000000	0x04f5dda4 - 0x00000000	0x04f5dda0 - 0x00000000	0x04f5dd9c - 0x00000000
0x04f5dd98 - 0x00000000	0x04f5dd94 - 0x00000000	0x04f5dd90 - 0x00000000	0x04f5dd8c - 0x00000000
0x04f5dd88 - 0x00000000	0x04f5dd84 - 0x00000000	0x04f5dd80 - 0x00000000	0x04f5dd7c - 0x00000000
0x04f5dd78 - 0x00000000	0x04f5dd74 - 0x00000000	0x04f5dd70 - 0x00000000	0x04f5dd6c - 0x00000000
0x04f5dd68 - 0x00000000	0x04f5dd64 - 0x00000000	0x04f5dd60 - 0x00000000	0x04f5dd5c - 0x00000000
0x04f5dd58 - 0x00000000	0x04f5dd54 - 0x00000000	0x04f5dd50 - 0x00000000	0x04f5dd4c - 0x00000000
0x04f5dd48 - 0x00000000	0x04f5dd44 - 0x00000000	0x04f5dd40 - 0x00000000	0x04f5dd3c - 0x00000000

```
0x04f5dd38 - 0x00000000    0x04f5dd34 - 0x00000000    0x04f5dd30 - 0x00000000    0x04f5dd2c - 0x00000000
0x04f5dd28 - 0x00000000    0x04f5dd24 - 0x00000000    0x04f5dd20 - 0x00000000    0x04f5dd1c - 0x00000000
0x04f5dd18 - 0x00000000    0x04f5dd14 - 0x00000000    0x04f5dd10 - 0x00000000    0x04f5dd0c - 0x00000000
0x04f5dd08 - 0x00000000    0x04f5dd04 - 0x00000000    0x04f5dd00 - 0x00000000    0x04f5dcfc - 0x00000000
0x04f5dcf8 - 0x00000000    0x04f5dcf4 - 0x00000000    0x04f5dcf0 - 0x00000000    0x04f5dcec - 0x00000000
0x04f5dce8 - 0x00000000    0x04f5dce4 - 0x00000000    0x04f5dce0 - 0x00000000    0x04f5dcdc - 0x00000000
0x04f5dcd8 - 0x00000000    0x04f5dcd4 - 0x00000000    0x04f5dcd0 - 0x00000000    0x04f5dccc - 0x00000000
0x04f5dcc8 - 0x00000000    0x04f5dcc4 - 0x00000000    0x04f5dcc0 - 0x00000000    0x04f5dcbc - 0x00000000
0x04f5dcb8 - 0x00000000    0x04f5dcb4 - 0x00000000    0x04f5dcb0 - 0x00000000    0x04f5dcac - 0x00000000
0x04f5dca8 - 0x00000000    0x04f5dca4 - 0x00000000    0x04f5dca0 - 0x00000000    0x04f5dc9c - 0x00000000

=== Task Control Block Info ===
Saved stack pointer      = 0x4f5eb10
Bottom of the stack      = 0x4f5ec38
Actual end of stack      = 0x4f5cc38
Stack Size               = 0x2000
Current stack usage      = 0x0
Maximum stack usage      = 0x1d58
Current stack margin     = 0x2a8
Most recent task error   = 0x3d0002
Delay/timeout ticks      = 0x0


=== Registers Info ===
   GPR (  0) - 0x41be0024
   GPR (  1) - 0x4f5e3c0   stack pointer, respond to %esp in ia32 architecture
   GPR (  3) - 0x41be0024
   GPR (  4) - 0x4fc05f8
   GPR (  5) - 0x1
   GPR (  6) - 0x28
   GPR (  7) - 0x4
   GPR (  8) - 0xc8
   GPR (  9) - 0x21b7fc
```

```
    GPR ( 10) - 0x8
    GPR ( 11) - 0x4fc6de0
    GPR ( 12) - 0x20000084
    GPR ( 22) - 0x1
    GPR ( 23) - 0x64
    GPR ( 25) - 0x65
    GPR ( 26) - 0x5
    GPR ( 27) - 0x77
    GPR ( 28) - 0x4f5e500
    GPR ( 29) - 0x4f5e512
    GPR ( 30) - 0x64
    GPR ( 31) - 0x4fc6de0

    Machine State Register - 0xb032
    Link Register          - 0x21d4a4   the return address of function
    Counter Register       - 0x21fd94
    Program Counter        - 0x219290
    Condition Register     - 0x20002084
    Fixed-point exception  - 0x20000000


=== Function Call Info ===
->vxTaskEntry + 0x54 (0x541548)
 ->cfm_agOlcEntry + 0x134 (0x21a82c)
  ->cfm_agSysWideTimerExpire + 0x160 (0x228dc8)
   ->cfm_agSMCCI_timerUpdate + 0x34 (0x222844)
    ->cfm_agSMCCI_dispatchEvent + 0x50 (0x22869c)
     ->cfm_agSMCCI_waiting + 0x10c (0x222a98)
      ->cfm_agSMCCI_enterState + 0x34 (0x2286e0)
       ->cfm_agSMCCI_dispatchEvent + 0x50 (0x22869c)
        ->cfm_agSMCCI_waiting + 0x78 (0x222a04)
         ->cfm_agXmitCCM + 0x18c (0x221d64)
          ->cfm_agSendNPPktMulticast + 0xe8 (0x2318d0)
```

```
            ->cfm_agIMEPEgressProcessing + 0x278 (0x21fda8)
             ->cfm_agCCReceiverBlk + 0x19c (0x221f24)
              ->cfm_agFindCliRmepByID + 0x48 (0x21daf0)
               ->cflSem + 0x246d124 (0x4161918)
                ->p2BtreeNext + 0x1c (0x219290)
PRESS any key for next record or 'q' to quit

value = 0 = 0x0
3540>
```

---

2. find the valuable information

From the above register information, we know that,

```
GPR (  0) - 0x41be0024
GPR (  1) - 0x4f5e3c0   stack pointer, respond to %esp in ia32 architecture
GPR (  3) - 0x41be0024
GPR (  4) - 0x4fc05f8
GPR ( 30) - 0x64
GPR ( 31) - 0x4fc6de0

Link Register          - 0x21d4a4   the return address of function
Program Counter        - 0x219290
```

2.1 the value of parameters: association and mepid

And from the disassemble code, we also find that the last modification to r30 and r31 is done in cfm_agFindCliRmepByID, shown below in purple background.
That is, (r31)=0x4fc6de0 is the pointer association, and (r30)=0x64=100 is mepid.

The last modification to r4 is done in cfm_agFindCliRmepByID too, shown below in purple background. That is, move the return value of cfm_agGetNextCliRmep to r4, rmep. So, rmep=0x4fc05f8.

dot1agCliRmep_t *cfm_agFindCliRmepByID(dot1agAssociation_t *association, uint16_t mepid)

```
OLCDV> l cfm_agFindCliRmepByID, 28
                        cfm_agFindCliRmepByID:

//save registers and allocate space
0x13fdbc  9421ffe0    stwu        r1,-32(r1)    /*(r1) = (r1) - 32, (r1) is 0x04f5e400, after this instruction, it will be 0x04f5e3e0,
                                                    Then, 0x04f5e3e0 will be the next callee's stack frame base pointer, and the callee is
                                                    cfm_agGetNextCliRmep
                                                  */
0x13fdc0  7c0802a6    mfspr       r0,LR         //move LR (Link Register, save return address of calling function) to r0, (LR)=0x00221f24

0x13fdc4  93c10018    stw         r30,24(r1)    //move (r30) to (r1)+24, that is, (r30)=0x04f5e512
0x13fdc8  93e1001c    stw         r31,28(r1)    //move (r31) to (r1)+28, that is, (r31)=0x04fc7c70
0x13fdcc  90010024    stw         r0,36(r1)     //move (r0) to (r1)+36, at this time, (r0)=0x00221f24, then, (r1)+36 content is 0x00221f24

//save parameters
0x13fdd0  7c7f1b78    or          r31,r3,r3     //(r3) => r31, r3 is the first parameter, that is, 'dot1agAssociation_t *association'
0x13fdd4  7c9e2378    or          r30,r4,r4     //(r4) => r30, r4 is the second parameter, that is, 'uint16_t mepid'

0x13fdd8  38800000    li          r4,0x0 # 0    //0 => r4, r4 is the local variable, rmep
0x13fddc  38600000    li          r3,0x0 # 0    //0 => r3

    if(association==NULL)
        return(NULL);
0x13fde0  2f9f0000    cmpi        crf7,0,r31,0x0 # 0            //r31 is the first parameter, association
0x13fde4  419e0030    bc          0xc,30, 0x13fe14 # 0x0013fe14  //return NULL; (r3)=NULL is the return value

0x13fde8  48000014    b           0x13fdfc # 0x0013fdfc
```

```
    while((rmep=cfm_agGetNextCliRmep(association, rmep))!=NULL)
    {
        if(rmep->mepid==mepid)
            return(rmep);
    }
//the following is the while-loop
0x13fdec  a0040016    lhz         r0,22(r4)           //(r4)+22 => r0, and the lowest 16 bits, that is, rmep->mepid => r0
0x13fdf0  7c832378    or          r3,r4,r4            //(r4) => r3, r4 is the local variable, rmep
0x13fdf4  7f80f000    cmp         crf7,0,r0,r30
0x13fdf8  419e001c    bc          0xc,30, 0x13fe14 # 0x0013fe14  //return rmep; (r3)=rmep is the return value

0x13fdfc  7fe3fb78    or          r3,r31,r31          //(r31) => r3, that is, the first parameter 'dot1agAssociation_t *association'
0x13fe00  4bfff981    bl          0x13f780 # cfm_agGetNextCliRmep
                                              /* rmep=fm_agGetNextCliRmep(association, rmep)
                                                 r3 is association, a pointer, the first parameter;
                                                 r4 is the local variable, rmep, and its initialization value is NULL, or
                                                 r4 is the return value rmep (by moveing r3 to r4 below)
                                               */

0x13fe04  7c641b78    or          r4,r3,r3            //the return value, rmep is in r3, then, copy it to r4
0x13fe08  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x13fe0c  409effe0    bc          0x4,30, 0x13fdec # 0x0013fdec

    return(NULL);
0x13fe10  38600000    li          r3,0x0 # 0          //if not found, move NULL to r3 as the return value

//restore registers and stack pointer
0x13fe14  80010024    lwz         r0,36(r1)           //(r1)+36 => r0, (r1)+36 is the return address
0x13fe18  7c0803a6    mtspr       LR,r0               //r0 => LR, the return address is saved into LR

0x13fe1c  83c10018    lwz         r30,24(r1)          //(r1)+24 => r30
0x13fe20  83e1001c    lwz         r31,28(r1)          //(r1)+28 => r31
0x13fe24  38210020    addi        r1,r1,0x20 # 32     //(r1)+32 => r1, that is, release the allocated stack space
```

```
0x13fe28  4e800020    blr                          //return
value = 0 = 0x0
```

2.2 the value of r3

Program Counter is 0x219290, we can see that program is suspended at 0x219290, line 8, shown in below codes.
Then, we can conclude that only line 2, 3, 5, 6 are executed. And only line 2 modified r0, only line 5 modified r3.
Since (r3)=0x41be0024, and it is the value after line 5 is executed, then, when calling p2BtreeNext, the first parameter nd=0x41be0016 (in r3);
that is, the initialization value of r3 is 0x41be0016, then, after line 2 is executed, (r0)=0x41be0024.

```
p2btree *p2BtreeNext(p2btree * nd)

3540> l p2BtreeNext, 29
                    p2BtreeNext:
1:  0x219274  9421fff0    stwu        r1,-16(r1)    //(r1)-16 => r1, then, r1=0x04f5e3c0, the stack pointer, and from here, we backtrace
2:  0x219278  80030008    lwz         r0,8(r3)      /*(r3) is the first parameter, p2btree* nd, (r3)+8 => r0, (r0)=0x41be0024=nd->b_right
                                                       Then, (r3)=0x41be0016, that is, nd=0x41be0016
                                                    */

    if (nd->b_right) {
3:  0x21927c  2f800000    cmpi        crf7,0,r0,0x0 # 0
4:  0x219280  419e002c    bc          0xc,30, 0x2192ac # 0x002192ac

        nd = nd->b_right;
5:  0x219284  80630008    lwz         r3,8(r3)            //(r3)+8 => r3, (r3)+8 is nd->b_right, then, new nd' is nd->b_right, saved in r3
6:  0x219288  2f830000    cmpi        crf7,0,r3,0x0 # 0   //'nd;' in for-loop, that is, nd!=0
7:  0x21928c  419e0020    bc          0xc,30, 0x2192ac # 0x002192ac

        for (; nd; nd = nd->b_left) {
            if (!nd->b_left)
                return (nd);
```

```
        }
8:  0x219290  80030004    lwz         r0,4(r3)      //(r3)+4 => r0, (r3)+4 is nd'->b_left
...
```

Appendix: An example of analyze a function

```
void cfm_agXmitCCM(dot1agMP_t *agMP)
```

(1) data structure and code

memory alignment is default 4bytes

```
typedef struct p2BalTree {
    struct p2BalTree    *b_bwd;
    struct p2BalTree    *b_left;
    struct p2BalTree    *b_right;
    signed char         b_bal;
} p2btree,*p2btree_pt;

Sizeof(p2btree) = 16

typedef struct dot1agDomain_s {

    p2btree *associationList;       /* list of associations (dot1agAssociation_t) */
    uint8_t     mdLevel;            /*  [20.7] */
    char        name[DOT1AG_MD_NAME_LEN+1];       /* support only char string format for now */
    //uint32_t nextMAID;       /* for MA mib table index */
    uint32_t    md_id;          /* for mib table index */

} dot1agDomain_t;
```

```
typedef struct dot1agMP_s
{
    p2btree               mpNode;       /* this node in p2btree */
    dot1agAssociation_t  *association;  /* back pointer */                 //offset=16
    portInstance_t        portInstance; /* uniquely identify a phy port */ //offset=20
    uint8_t               macAddress[6];                                  //offset=24
    uint8_t              type;                                            //offset=30
    union
    {
        dot1agMEP_t   mep;
        dot1agMIP_t   mip;
    } mp;

} dot1agMP_t;

void cfm_agXmitCCM(dot1agMP_t *agMP)
{
    uint32_t  bytesLeft;
    uint8_t   buffer[DOT1AG_MAX_MTU_SIZE];

#if 1
    CFM_CCMTRACE("xmit CCM -- port=0x%x type=%d ma=%d md=%d mepid=%d",
        agMP->portInstance, agMP->type,
        agMP->association->ma_id,
        agMP->association->domain->mdLevel,
        agMP->mp.mep.mepid);
#endif

    if ((!agMP->mp.mep.MEPactive) || (!agGlobal.adminState))
    {
        CFM_DEBUG("[%s-%d]: no CCM sent; MEP [%d] not active",
                          __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
        return;
```

```
    }

    if((memcmp(agMP->macAddress, nullMAC, 6)==0)  &&
        (!cfm_agGetPortMac(agMP->portInstance, agMP->macAddress)))
    {
        CFM_DEBUG("[%s-%d]: MEP [%d] MAC address is NULL; no CCM pkt sent",
                            __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
        return;
    }

    if((memcmp(agMP->macAddress, bcastMAC, 6)==0)  &&
        (!cfm_agGetPortMac(agMP->portInstance, agMP->macAddress)))
    {
        CFM_DEBUG("[%s-%d]: MEP [%d] MAC address is bcast; no CCM pkt sent",
                            __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
        return;
    }

#ifdef ITU_Y1731_SUPPORT
    /* zero RDICondition */
    agMP->mp.mep.mepDefect.RDICondition = 0;
#endif

    memset(buffer, 0, sizeof(buffer));
    bytesLeft=cfm_agCreateCfmCCMPdu(agMP, buffer, sizeof(buffer));

    /*
     * Send PDU out of the correct port(s)
     */
    /* if port is not reachable, don't send pkt anyway */
    if ((agMP->type == DOT1AG_MP_IMEP)  &&
         (!cfm_agIsMPConnected(agMP->portInstance)))
         return;
```

```
#ifdef ITU_Y1731_NP_TAG_PKT
    cfm_agSendNPPktMulticast(agMP, buffer, sizeof(buffer)-bytesLeft, FALSE, DOT1AG_PDU_CCM_PKT);
#else
    cfm_agSendPktMulticast(agMP, buffer, sizeof(buffer)-bytesLeft, FALSE);
#endif

    /* update stats */
    agMP->mp.mep.mepStats.ccmSent++;
}
```

(2) dissemble code and analysis

```
3540> l cfm_agXmitCCM, 108
                      cfm_agXmitCCM:

//save registers and allocate space
0x221bd8  9421fa00    stwu        r1,-1536(r1)     //(r1) = > (r1)-1536, allocate space of 1536 bytes
0x221bdc  7c0802a6    mfspr       r0,LR            //move (LR) to r0, the return address of calling cfm_agXmitCCM
0x221be0  93a105f4    stw         r29,1524(r1)     //(r29) => (r1)+1524
0x221be4  93e105fc    stw         r31,1532(r1)     //(r31) => (r1)+1532
0x221be8  90010604    stw         r0,1540(r1)      //r0 (the return address) => (r1)+1540

//save parameters
0x221bec  7c7f1b78    or          r31,r3,r3        /*(r3)'s initial value is the first parameter of void cfm_agXmitCCM(dot1agMP_t *agMP)
                                                      That is, agMP, a pointer to dot1agMP_t object
                                                      (r3) => (r31), then, (r31) is agMP (the pointer), that is, r31 saves the parameter
                                                  */

    CFM_CCMTRACE("xmit CCM -- port=0x%x type=%d ma=%d md=%d mepid=%d",
        agMP->portInstance, agMP->type,
        agMP->association->ma_id,
        agMP->association->domain->mdLevel,
```

```
        agMP->mp.mep.mepid);
0x221bf0  88a3001e    lbz         r5,30(r3)        //(r3)+30 => r5, and the lowest 1 byte, that is, agMP->type
0x221bf4  81630010    lwz         r11,16(r3)       //(r3)+16 => r11, that is, agMP->association
0x221bf8  81230010    lwz         r9,16(r3)        //(r3)+16 => r9, that is, agMP->association
0x221bfc  81290018    lwz         r9,24(r9)        //(r9)+24 => r9, that is, agMP->association->domain
0x221c00  88e90004    lbz         r7,4(r9)         //(r9)+4  => r7, and the lowest 1 byte, that is, agMP->association->domain->mdLevel
0x221c04  a1030026    lhz         r8,38(r3)        //(r3)+38 => r8, and the lowest 2 bytes, that is, agMP->mp.mep.mepid
0x221c08  3c60006d    lis         r3,0x6d # 109
0x221c0c  386369e4    addi        r3,r3,0x69e4 # 27108  //the string address
0x221c10  809f0014    lwz         r4,20(r31)       //(r31)+20 => r4, that is, agMP->portInstance
0x221c14  80cb001c    lwz         r6,28(r11)       //(r11)+28 => r6, that is, agMP->association->ma_id
0x221c18  4cc63182    crxor       crb6,crb6,crb6
0x221c1c  4bff9069    bl          0x21ac84 # CFM_CCMTRACE

0x221c20  881f0038    lbz         r0,56(r31)
0x221c24  70090080    andi.       r9,r0,0x80
0x221c28  41820014    bc          0xc,2, 0x221c3c # 0x00221c3c
0x221c2c  3d2000ed    lis         r9,0xed # 237
0x221c30  880952fc    lbz         r0,21244(r9)
0x221c34  2f800000    cmpi        crf7,0,r0,0x0 # 0
0x221c38  409e0020    bc          0x4,30, 0x221c58 # 0x00221c58

    CFM_DEBUG("[%s-%d]: no CCM sent; MEP [%d] not active",
                            __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
0x221c3c  a0df0026    lhz         r6,38(r31)           //(r3)+38 => r6, and the lowest 2 bytes, that is, agMP->mp.mep.mepid
0x221c40  3c60006d    lis         r3,0x6d # 109
0x221c44  38636a18    addi        r3,r3,0x6a18 # 27160  //the string address
0x221c48  3c80006d    lis         r4,0x6d # 109
0x221c4c  38846a44    addi        r4,r4,0x6a44 # 27204  //__FUNCTION__
0x221c50  38a0002c    li          r5,0x2c # 44          //__LINE__
0x221c54  480000a0    b           0x221cf4 # 0x00221cf4 //call CFM_DEBUG

    if((memcmp(agMP->macAddress, nullMAC, 6)==0)  &&
```

```
       (!cfm_agGetPortMac(agMP->portInstance, agMP->macAddress)))
0x221c58  3bbf0018    addi        r29,r31,0x18 # 24     //(r31)+24 => r29, that is, agMP->macAddress
0x221c5c  7fa3eb78    or          r3,r29,r29            //(r29) => r3
0x221c60  3c8000a0    lis         r4,0xa0 # 160
0x221c64  388460e4    addi        r4,r4,0x60e4 # 24804  //nullMAC
0x221c68  38a00006    li          r5,0x6 # 6            //6 => r5
0x221c6c  48331fb1    bl          0x553c1c # memcmp

0x221c70  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x221c74  409e0034    bc          0x4,30, 0x221ca8 # 0x00221ca8
0x221c78  807f0014    lwz         r3,20(r31)            //(r31)+20 =>  r3, that is, agMP->portInstance
0x221c7c  7fa4eb78    or          r4,r29,r29            //(r29) => r4, that is, agMP->macAddress
0x221c80  4bff9ed5    bl          0x21bb54 # cfm_agGetPortMac

0x221c84  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x221c88  40be0020    bc          0x5,30, 0x221ca8 # 0x00221ca8

       CFM_DEBUG("[%s-%d]: MEP [%d] MAC address is NULL; no CCM pkt sent",
                          __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
0x221c8c  a0df0026    lhz         r6,38(r31)
0x221c90  3c60006d    lis         r3,0x6d # 109
0x221c94  38636a54    addi        r3,r3,0x6a54 # 27220
0x221c98  3c80006d    lis         r4,0x6d # 109
0x221c9c  38846a44    addi        r4,r4,0x6a44 # 27204
0x221ca0  38a00034    li          r5,0x34 # 52
0x221ca4  48000050    b           0x221cf4 # 0x00221cf4 //call CFM_DEBUG

    if((memcmp(agMP->macAddress, bcastMAC, 6)==0)  &&
       (!cfm_agGetPortMac(agMP->portInstance, agMP->macAddress)))
0x221ca8  3bbf0018    addi        r29,r31,0x18 # 24
0x221cac  7fa3eb78    or          r3,r29,r29
0x221cb0  3c800081    lis         r4,0x81 # 129
0x221cb4  3884d190    addi        r4,r4,0xd190 # -11888
```

```
0x221cb8  38a00006    li          r5,0x6 # 6
0x221cbc  48331f61    bl          0x553c1c # memcmp

0x221cc0  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x221cc4  409e003c    bc          0x4,30, 0x221d00 # 0x00221d00
0x221cc8  807f0014    lwz         r3,20(r31)
0x221ccc  7fa4eb78    or          r4,r29,r29
0x221cd0  4bff9e85    bl          0x21bb54 # cfm_agGetPortMac

0x221cd4  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x221cd8  40be0028    bc          0x5,30, 0x221d00 # 0x00221d00

        CFM_DEBUG("[%s-%d]: MEP [%d] MAC address is bcast; no CCM pkt sent",
                                __FUNCTION__,__LINE__,agMP->mp.mep.mepid);
0x221cdc  a0df0026    lhz         r6,38(r31)
0x221ce0  3c60006d    lis         r3,0x6d # 109
0x221ce4  38636a8c    addi        r3,r3,0x6a8c # 27276
0x221ce8  3c80006d    lis         r4,0x6d # 109
0x221cec  38846a44    addi        r4,r4,0x6a44 # 27204
0x221cf0  38a0003c    li          r5,0x3c # 60
0x221cf4  4cc63182    crxor       crb6,crb6,crb6
0x221cf8  4bff8eb9    bl          0x21abb0 # CFM_DEBUG

0x221cfc  48000074    b           0x221d70 # 0x00221d70

0x221d00  38000000    li          r0,0x0 # 0
0x221d04  981f06c1    stb         r0,1729(r31)

    memset(buffer, 0, sizeof(buffer));
0x221d08  3ba10010    addi        r29,r1,0x10 # 16   //(r1)+16 => r29, that is, 0x04f5e4f0 + 0x10 = 0x04f5e500
0x221d0c  7fa3eb78    or          r3,r29,r29         /*(r29) => r3, the first parameter, that is, buffer (the start address)
                                                       Then, we can say that the offset of buffer in the stack is 0x10=16
                                                     */
```

```
0x221d10  38800000    li          r4,0x0 # 0          //0 => r4
0x221d14  38a005dc    li          r5,0x5dc # 1500     //1500 => r5
0x221d18  48331fdd    bl          0x553cf4 # memset

    bytesLeft=cfm_agCreateCfmCCMPdu(agMP, buffer, sizeof(buffer));
0x221d1c  7fe3fb78    or          r3,r31,r31          //(r31) => r3, the first parameter, that is, agMP
0x221d20  7fa4eb78    or          r4,r29,r29          //(r29)=buffer => r4
0x221d24  38a005dc    li          r5,0x5dc # 1500     //1500 => r5
0x221d28  4bfff5dd    bl          0x221304 # cfm_agCreateCfmCCMPdu

0x221d2c  7c7d1b78    or          r29,r3,r3           //the return value of calling cfm_agCreateCfmCCMPdu, that is, bytesLeft => r3 => r29
0x221d30  881f001e    lbz         r0,30(r31)
0x221d34  2f800001    cmpi        crf7,0,r0,0x1 # 1
0x221d38  40be0014    bc          0x5,30, 0x221d4c # 0x00221d4c

        (!cfm_agIsMPConnected(agMP->portInstance)))
0x221d3c  807f0014    lwz         r3,20(r31)          //the first parameter
0x221d40  4bff9db5    bl          0x21baf4 # cfm_agIsMPConnected

    cfm_agSendNPPktMulticast(agMP, buffer, sizeof(buffer)-bytesLeft, FALSE, DOT1AG_PDU_CCM_PKT);
0x221d44  2f830000    cmpi        crf7,0,r3,0x0 # 0
0x221d48  419e0028    bc          0xc,30, 0x221d70 # 0x00221d70

0x221d4c  20bd05dc    subfic      r5,r29,0x5dc # 1500  //1500-(r29) => r5, (r29) is bytesLeft, the return value
0x221d50  7fe3fb78    or          r3,r31,r31          //agMP
0x221d54  38810010    addi        r4,r1,0x10 # 16     //(r1)+16 => r4, that is, 0x04f5e4f0 + 0x10 = 0x04f5e500, the buffer's start address
0x221d58  38c00000    li          r6,0x0 # 0          //FALSE=0
0x221d5c  38e00005    li          r7,0x5 # 5          //DOT1AG_PDU_CCM_PKT=5
0x221d60  4800fa89    bl          0x2317e8 # cfm_agSendNPPktMulticast

   agMP->mp.mep.mepStats.ccmSent++;
0x221d64  813f0684    lwz         r9,1668(r31)   //(r31)+1668 => r9, that is, agMP->mp.mep.mepStats.ccmSent
```

```
0x221d68  39290001    addi        r9,r9,0x1 # 1  //(r9)+1 = > r9
0x221d6c  913f0684    stw         r9,1668(r31)   //(r9) => (r31)+1668

//restore stack and registers
0x221d70  80010604    lwz         r0,1540(r1)    //(r1)+1540 => r0, that is , the return address of calling cfm_agXmitCCM
0x221d74  7c0803a6    mtspr       LR,r0          //(r0) => LR, the return address
0x221d78  83a105f4    lwz         r29,1524(r1)   //(r1)+1524 => r29
0x221d7c  83e105fc    lwz         r31,1532(r1)   //(r1)+1532 => r31
0x221d80  38210600    addi        r1,r1,0x600 # 1536  //(r1)+1536 => (r1)
0x221d84  4e800020    blr                            //return, if there is a return value, it is in r3
value = 0 = 0x0
```