

B5G_UE - 错误 #3835

MSG5状态后pdcch漏检

2025-07-30 09:54 - 张倩

状态:	已解决	开始日期:	2025-07-30
优先级:	普通	计划完成日期:	
指派给:	张倩	% 完成:	0%
类别:		预期时间:	0.00 小时
目标版本:		耗时:	0.00 小时
描述			
MSG5状态后pdcch漏检			

历史记录

#1 - 2025-07-30 11:04 - 张倩

- 文件 0210bea4-609f-4d10-8ba4-5724da654f2d.jpeg 已添加
- 文件 4dc021c3-fd07-4871-a93c-1d9e6d3478eb.jpeg 已添加
- 文件 ce91b795-9d33-4659-a196-a13409644e1d.jpeg 已添加
- 文件 df93a32b-acfd-41ae-ba40-cb66e16d0893.jpeg 已添加

msg5状态之前 startcce使用的是红框参数start_cce_position
msg5状态之后 startcce使用的是绿框参数start_cce_position1[]

```
3:
4: typedef struct searchSpaceCandidate
5: {
6:     bool valid_flag ; // 0- invalid , 1- valid
7:     uint8_t start_symbol_idx ; // 起始符号
8:     uint8_t start_cce_position;
9:     uint8_t start_cce_position1[20]; // 256 414 coreset 1
10:    uint8_t cce_num ;
11:    uint8_t dci_format;
12:    uint16_t dci_size;
13:    uint32_t rnti ;
14: } SearchSpaceCandidate;
15:
```

原代码中只在pdcch_dci_remap_fun函数中在msg5之后将start_cce_position替换成了start_cce_position1[]，这个函数外面其他startcce用的仍然是start_cce_position，但是msg5之后start_cce_position没赋值，默认是0，导致pdcch漏检了一些候选集；

```

694:
695: static int16_t pdcch_dci_remap_fun(uint16_t coresetIdx, uint16_t ssIdx, uint16_t candIdx,
696:     float *sigma2Ret, float *sigpowRet, float *snrDbRet )
697: {
698:     //StepCONFIG_PARAMS pConfigParams = phy_gue_main_get_ctx();
699:     //uint32_t nCtxNum = get_d1_sf_ctx(nSlotIdx);
700: #if 0
701:     PHYStateDLRX *pPHYStateRx = phy_gue_d1_get_ctx();
702: #endif
703:     struct timespec Starttime;
704:     struct timespec Endtime;
705:     // clock_gettime(CLOCK_REALTIME, &Starttime);
706:
707:     //pdcchRxStruct * pPdcchRxStruct_dm_ptr = &d_psPHYStateRx->sPdcchRxStruct;
708:
709:     SteCoreSet * pCoreset = ( (SteCoreSet *)pPdcchRxStruct_dm_ptr->pCsTbl + coresetIdx );
710:
711:     SearchSpaceCandidate * pSSCandidate = &pPdcchRxStruct_dm_ptr->ssCandidate[coresetIdx][ssIdx][candIdx];
712: #if 0
713:     int32_t nRxAntNum = LOAD_EX_W(&pPHYStateRx->nRxAntNum);
714: #endif
715: #if 1
716:     int32_t nRxAntNum = 1;
717: #endif
718:     uint16_t * pCsRbIdxTbl = (uint16_t *)pPdcchRxStruct_dm_ptr->pCsRbIdx[coresetIdx];
719:     uint16_t nCsRbNum = pPdcchRxStruct_dm_ptr->nCsRbNum[coresetIdx];
720:
721:     //
722:     struct bblib_pdcch_determine_phy_rbs_request pdcchRbsRequest;
723:     struct bblib_pdcch_determine_phy_rbs_response pdcchRbsResponse;
724:     uint8_t slot = get_rx_nr_slot();
725:
726:     pdcchRbsRequest.cce_reg_mapping_type_cs = pCoreset->cce2RegMapTyp;
727:     pdcchRbsRequest.interleave_size_R_cs = pCoreset->intrlvrRows;
728:     pdcchRbsRequest.n_duration_cs = pCoreset->duration;
729:     pdcchRbsRequest.n_reg_dundle_size_cs = pCoreset->regBndlSz;
730:     pdcchRbsRequest.n_shift_cs = pCoreset->shiftIndex;
731:     pdcchRbsRequest.start_cce_idx_pdcch = pSSCandidate->start_cce_position;
732:     pdcchRbsRequest.n_agg_level_pdcch = pSSCandidate->cce_num;
733:     pdcchRbsRequest.rb_idx_bwp_cs = pCsRbIdxTbl;
734:     pdcchRbsRequest.n_rb_cs = nCsRbNum;
735:
736:     pdcchRbsResponse.rb_idx_bwp_pdcch = (uint16_t *) (pCsRbIdxTbl + RUP64B(MAX_NUM_OF_PRB_IN_FULL_BAND) );
737:     pdcchRbsResponse.rb_bitmap_pdcch = (uint8_t *) (pCsRbIdxTbl + RUP64B(MAX_NUM_OF_PRB_IN_FULL_BAND)*2 );
738:     uint16_t ueState;
739:     ueState = LOAD_EX_W(&ueCfg->ueState);
740:     if(ueState >= UE_MSG5_SS_CFG_SUCC)
741:     {
742:         pdcchRbsRequest.start_cce_idx_pdcch = pSSCandidate->start_cce_position1[g_pdcch_freq_data_hdr_dm_ptr->slot];
743:     }
744:
745:     uint8_t ueType = LOAD_EX_W(&ueCfg->ueType);
746:

```

e_pdcch_func.s.c

symbol Name (Alt+L)

- PALLADIUM_DEBUG_MEMCPY
- (anons_uint16_2byte)
 - lowByte
 - highByte
- s_uint16_2byte
- (anons_uint16_2byte)
 - uin16Acc
 - byteAcc
- u_uint16_2byte
- trigger_pdcch
- uDiCnt
- if 0
- endif
- c_pdcch_polar_decode_P
- c_pdcch_polar_decode_P_assist
- c_pdcch_polar_decode_bitmask_lut
- INTERLEAVING_PATTERN
- if 0
- endif
- if 1
- pPdcchCoresetTbl
- cnt
- PDCCH_QFTmpCalc
- endif
- PDCCH_DecodeParamCalc
- bblib_pdcch_determine_coreset_rbs
- pdcch_coreset_tbl_fun
- pdcch_dci_remap_dummy_fun
- pdcch_dci_remap_fun
- bblib_pdcch_extract_coreset_rbs
- pdcch_coreset_rbl_cal
- pdcch_coreset_demod_fun
- pdcch_dci_decode_fun
- phy_gue_d1_pdcch_func

```

2378:     int8_t skipFlg[CANDIDATE_NUM_PER_SS] = {0};
2379:
2380:     while( candCnt < nCandidates )
2381:     {
2382:         for( curCandIdx=0; curCandIdx < nCandidates; curCandIdx++ )
2383:         {
2384:             if( 0==skipFlg[curCandIdx] )
2385:             {
2386:                 curStartCce = (pSSCandidate+curCandIdx)->start_cce_position;
2387:                 curAgglvl = (pSSCandidate+curCandIdx)->cce_num;
2388:                 skipFlg[curCandIdx] = 1;
2389:                 candBlindDetSeq[candCnt] = curCandIdx;
2390:                 candCnt++;
2391:                 break;
2392:             }
2393:         }
2394:         for( ; curCandIdx < nCandidates; curCandIdx++ )
2395:         {
2396:             if( (0==skipFlg[curCandIdx]) && (curStartCce == ((pSSCandidate+curCandIdx)->start_cce_position)) && ( curAgglvl == ((pSSCandidate+curCandIdx)->cce_num) ) )
2397:             {
2398:                 skipFlg[curCandIdx] = 1;
2399:                 candBlindDetSeq[candCnt] = curCandIdx;
2400:                 candCnt++;
2401:             }
2402:         }
2403:     } // end while candCnt < nCandidates
2404:
2405:     //
2406:     preBlindDetectType = dcType;
2407:     // end if preBlindDetectType==d...
2408:
2409:     uint8_t excludedCceBitMap[32]={0};
2410:     //memset(excludedCceBitMap, 0, sizeof(excludedCceBitMap));
2411:     uDiCnt = 0;
2412:     for( candIdx=0; candIdx < nCandidates; candIdx++ )
2413:     {
2414:         pPdcchRxStruct_dm_ptr->nDiBlindDetectCnt++;
2415:         LOG_ERROR_S("[ZQ nCandidates] candIdx:%d nDiBlindDetectCnt:%d\n", candIdx, pPdcchRxStruct_dm_ptr->nDiBlindDetectCnt);
2416:         pSSCandidateCur = pSSCandidate + candBlindDetSeq[candIdx];
2417:         pSSCandidatePre = pSSCandidate + candBlindDetSeq[(candIdx-1)>0];
2418:         uint8_t m256CandBitMap[32]={0};
2419:         //memset(m256CandBitMap, 0, sizeof(m256CandBitMap));
2420:
2421:         uint16_t * pm256CandBitMap = (uint16_t *) (m256CandBitMap[0]);
2422:         uint32_t candBitMap = (1<<(pSSCandidateCur->cce_num))-1;
2423:         candBitMap = candBitMap << (pSSCandidateCur->start_cce_position % 16 );
2424:         uint16_t * pu32CandBitMap = (uint16_t *) (&candBitMap);
2425:         *(pm256CandBitMap + (pSSCandidateCur->start_cce_position / 16) ) = *pu32CandBitMap;
2426:         *(pm256CandBitMap + (pSSCandidateCur->start_cce_position / 16) + 1) = *(pu32CandBitMap + 1);
2427:
2428:         uint8_t m256CandBitMapMasked[32];
2429:         for( int i = 0; i < 32; i++) {
2430:             m256CandBitMapMasked[i] = excludedCceBitMap[i] & m256CandBitMap[i];
2431:         }

```

修改点：msg5状态之后，在pdcch任务入口处就把start_cce_position1[]赋值给start_cce_position

```
22421 - case UE_MSG5_SS_CFG_SUCC:
22422 /*
22423  fix bug: 特殊场景考虑如下描述
22424  当UE处于UE_MSG5_SS_CFG_SUCC这个状态的时候,表示msg4 DCI/pdsch都解对了且收到DL的msg4配置了
22425  UE反馈了MSG4 ACK但是基站可能没有解对,然后会发起msg4的重传,所以UE这个状态需要盲检msg4重传的DCI1_0
22426 */
22427
22428 #if 1
22429 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].candidateNum=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].candidateNum);
22430 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].coresetIdx=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].coresetIdx);
22431 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].searchSpaceIdx=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].searchSpaceIdx);
22432 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].startFlag=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].startFlag);
22433 #endif
22434 uint32_t cRnti;
22435 cRnti= LOAD_EX_W(&ueCfg->cRnti);
22436 //LOG_ERROR_S("UE_MSG5_SS_CFG_SUCC cRnti: %d \n",cRnti);
22437 for( int8_t csIdx=0; csIdx<MAX_CORESET_NUM; csIdx++)
22438 for( int8_t ssIdx=0; ssIdx<MAX_SEARCH_SPACE_NUM; ssIdx++)
22439 {
22440 for( int8_t candIdx=0; candIdx<CANDIDATE_NUM_PER_SS; candIdx++ )
22441 {
22442 pdcchRxstruct_dm_ptr->ssCandidate[csIdx][ssIdx][candIdx].rnti = cRnti;
22443 pdcchRxstruct_dm_ptr->ssCandidate[csIdx][ssIdx][candIdx].start_cce_position = pPdcchRxstruct_dm_ptr->ssCandidate[csIdx][ssIdx][candIdx].start_cce_position1[g_pdcch_freq_data_hdr_dm_ptr->slot];
22444 }
22445 break;
22446 }
22447
22448 - case UE_MSG5_DCI_PARSE_SUCC: // 当处于这个状态的时候,有可能需要去盲检msg5重传的DCI0_1
22449 #if 0
22450 ape_csu_dma_id_G2L_ch2ch3_transfer((uint64_t)&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5],
22451 (uint64_t)DM_TO_CSU_ADOR(&pPdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5]),
22452 sizeof(BlindDetectInfo),
22453 DMA_TAG_G2L,
22454 1);
22455 #endif
22456 #if 1
22457 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].candidateNum=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].candidateNum);
22458 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].coresetIdx=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].coresetIdx);
22459 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].searchSpaceIdx=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].searchSpaceIdx);
22460 pdcchRxstruct_dm_ptr->blindDetect[BLIND_DETECT_DCI_MSG5].startFlag=LOAD_EX_B(&ueCfg->blindDetect[BLIND_DETECT_DCI_MSG5].startFlag);
22461 for( int8_t csIdx=0; csIdx<MAX_CORESET_NUM; csIdx++)
22462 for( int8_t ssIdx=0; ssIdx<MAX_SEARCH_SPACE_NUM; ssIdx++)
22463 {
22464 for( int8_t candIdx=0; candIdx<CANDIDATE_NUM_PER_SS; candIdx++ )
22465 {
22466 pdcchRxstruct_dm_ptr->ssCandidate[csIdx][ssIdx][candIdx].start_cce_position = pPdcchRxstruct_dm_ptr->ssCandidate[csIdx][ssIdx][candIdx].start_cce_position1[g_pdcch_freq_data_hdr_dm_ptr->slot];
22467 }
22468 }
22469 #endif
22470 - case UE_ACTIVE:
```

#2 - 2025-07-30 11:04 - 张倩

- 状态从新建变更为已解决

文件

0210bea4-609f-4d10-8ba4-5724da654f2d.jpeg	49.4 KB	2025-07-30	张倩
4dc021c3-fd07-4871-a93c-1d9e6d3478eb.jpeg	272 KB	2025-07-30	张倩
ce91b795-9d33-4659-a196-a13409644e1d.jpeg	388 KB	2025-07-30	张倩
df93a32b-acfd-41ae-ba40-cb66e16d0893.jpeg	337 KB	2025-07-30	张倩