1. 问题描述

Jan 1 08:00:20 ubuntu kernel: [ 22.674331] fec 2188000.ethernet eth0: MDIO read timeout

Jan 1 08:00:21 ubuntu kernel: [ 23.704330] fec 2188000.ethernet eth0: MDIO read timeout

Jan 1 08:00:21 ubuntu kernel: [ 23.734347] fec 2188000.ethernet eth0: MDIO write timeout

Jan 1 08:00:22 ubuntu kernel: [ 24.764322] fec 2188000.ethernet eth0: MDIO read timeout

Jan 1 08:00:29 ubuntu kernel: [ 31.124298] mmc3: Timeout waiting for hardware interrupt.

出现之后网口不通。

1. 问题定位

查询phy是一个延时工作队列。如下，在查询phy的状态

INIT\_DELAYED\_WORK(&dev->state\_queue, phy\_state\_machine);

queue\_delayed\_work(system\_power\_efficient\_wq, &phydev->state\_queue,PHY\_STATE\_TIME \* HZ);

phy\_state\_machine-> fec\_enet\_mdio\_read

fec\_enet\_mdio\_read

writel

wait\_for\_completion\_timeout(&fep->mdio\_done,usecs\_to\_jiffies(FEC\_MII\_TIMEOUT))

fep->mdio\_done准备好是靠中断判断

fec\_enet\_interrupt

complete(&fep->mdio\_done);

上述产生timeout的流程清楚了，可能产生的原因：

1. 可能是丢中断。
2. 可能是中断没执行到。

Mac的数据传送完成中断标志是写清，不太可能丢中断。第二个就是中断没有执行到。如果按照正常内核对中断的处理，这也不太可能。后来查资料发现实时补丁会把中断线程化。

这样就解释的通了，中断线程有可能没有及时处理。按照这个设想继续分析。

中断线程化。

一般的中断处理是分为上半部和下半部。上半部处理紧急事务，下半部处理耗时操作。顶半部和底半部最大不同在于，下半部是可以被其他中断打断的，这样就不会耽误其他中断的进行了。在linux内核中，中断是最优先处理的。它可以打断其他进程执行。

虽然上半部处理紧急任务，耗时不会太大。但也会有一定影响。会关中断

打了实时补丁的内核会强制把中断线程化。把中断处理当中线程，有内核统一调度，以保证高优先级的进程优先执行。其实也是分了上半部和下半部，上半部直接返回IRQ\_WAKE\_THREAD，唤醒对应的中断线程。由中断线程处理所有中断处理操作。

ps -lax

1 0 82 2 -51 - 0 0 irq\_th S ? 0:01 [irq/284-sdma]

1 0 83 2 -51 - 0 0 irq\_th S ? 0:00 [irq/24-2020000.]

1 0 84 2 -51 - 0 0 irq\_th S ? 0:00 [irq/297-21e8000]

1 0 85 2 -51 - 0 0 irq\_th S ? 0:00 [irq/298-21ec000]

1 0 86 2 -51 - 0 0 irq\_th S ? 0:00 [irq/299-21f0000]

1 0 87 2 -51 - 0 0 irq\_th S ? 0:00 [irq/300-21f4000]

1 0 88 2 -53 - 0 0 irq\_th S ? 0:00 [irq/287-2188000]

1 0 89 2 -53 - 0 0 irq\_th S ? 0:00 [irq/288-2188000]

1 0 101 2 -51 - 0 0 irq\_th S ? 0:00 [irq/269-20bc000]

1 0 102 2 -51 - 0 0 irq\_th S ? 0:09 [irq/289-mmc1]

1 0 103 2 -52 - 0 0 irq\_th S ? 0:11 [irq/289-s-mmc1]

1 0 105 2 -51 - 0 0 irq\_th S ? 0:00 [irq/290-mmc2]

1 0 106 2 -52 - 0 0 irq\_th S ? 0:00 [irq/290-s-mmc2]

1 0 107 2 -51 - 0 0 irq\_th S ? 0:00 [irq/67-2198000.]

1 0 108 2 -51 - 0 0 irq\_th S ? 0:08 [irq/291-mmc3]

1 0 109 2 -52 - 0 0 irq\_th S ? 0:00 [irq/291-s-mmc3]

wait\_for\_completion\_timeout里面调用schedule\_timeout，让出cpu让其他线程执行。schedule\_timeout本质是个定时器。在调度中有很高的优先级。

现在有这样的情况，让出cpu之后，有一个最高的实时进程运行。高实时进程运行完之后让出cpu.此时可能有多个同优先级的中断线程，线程都是FIFO的调度模式，可能cpu调度了mac中断之外的其他线程。此时wait\_for\_completion\_timeout几经超时了。在调度到这个工作队列的时候就返回超时了。

为了验证是因为mac中断的得不到及时的调度。更容易的复现该问题。把所用中断线程的优先级都调低，调到一般线程的优先级。并将把mdio read的timeout时间调小一点。

把jdagv的实时进程优先级调到最高，最高实时进程里面死循环执行printf和LOG\_WAN

这样的确会比较容易的复现问题。

1. 问题解决

1，修改mac中断的优先级，比其他中断线程优先级高2个优先级

/home/jhr/jhr/linuxkernel/huawei/rtlinux2/kernel/irq/ manage.c

setup\_irq\_thread(struct irqaction \*new, unsigned int irq, bool secondary)

2、fec\_enet\_mdio\_read/fec\_enet\_mdio\_write修改超时时间FEC\_MII\_TIMEOUT 30000🡪4000000

3、fec\_enet\_mdio\_read/fec\_enet\_mdio\_write注释掉 return -ETIMEDOUT