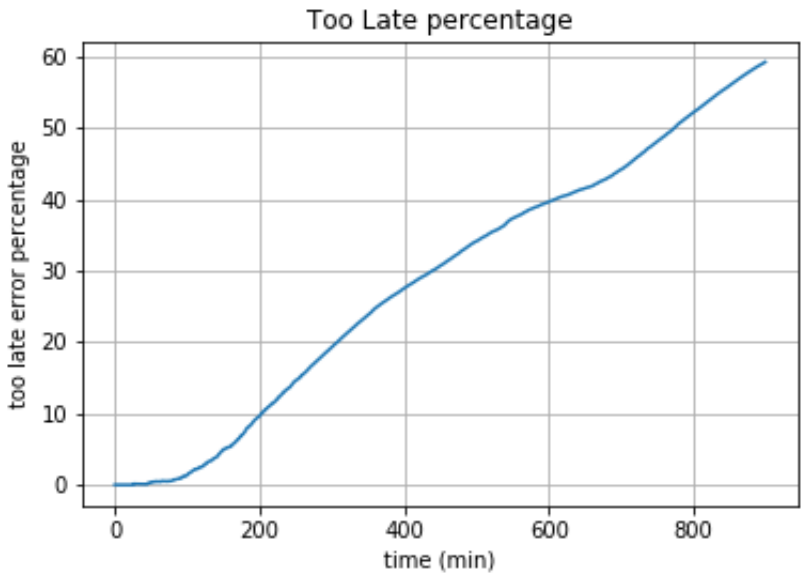
多终端大量上行测试的丢包分析

葛鑫 2019-03-05

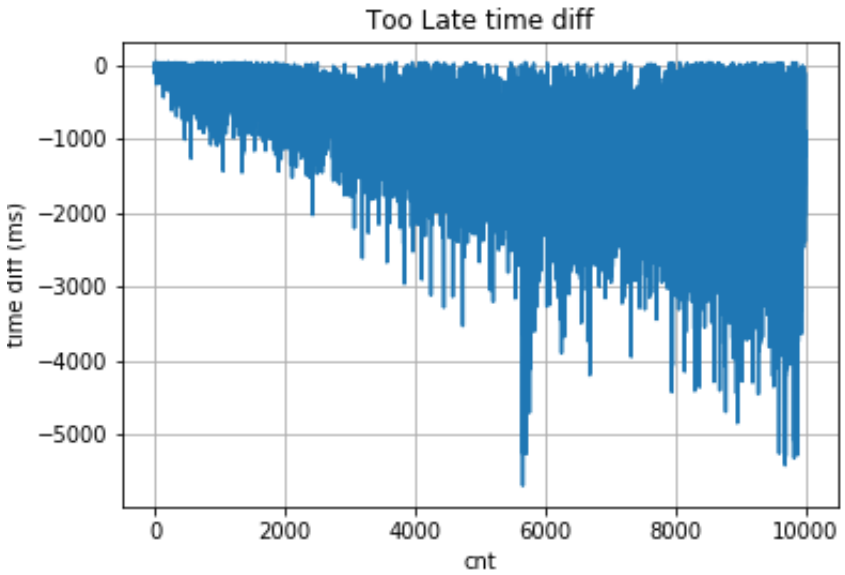
现象

使用多个终端并大量发送(间隔10s)上行包的情况下，丢包率随时间越来越大，几个小时后系统完全无法正常工作。具体表现为：

丢包都是**网关too late**下行拒绝



因此，对每一对相邻的上行`json up`和下行`json down`，分析了包**传给服务器的时间**`t1`和**从服务器拿到下行的时间**`t2`，计算差值，结果如下。纵坐标是`t2-t1`。



可以看到，时间差也随着时间越来越大，并且都是负值。

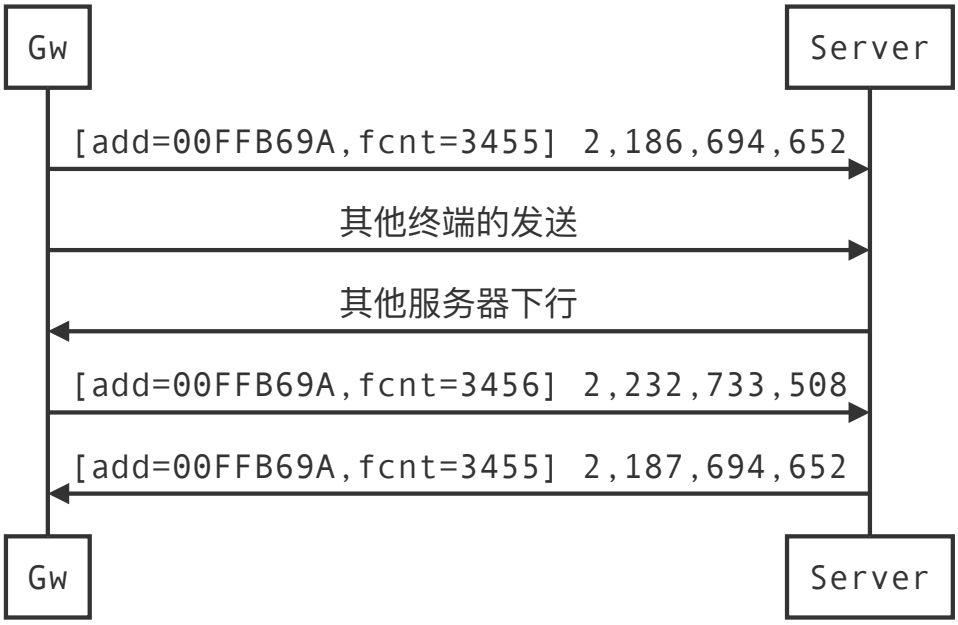
原因

为分析原因，找出了网关打印数据分析

```
JSON up: {"rxpk":[{"tmst":2232733508,"time":"2019-01-24T13:04:39.782719000Z","tmms":1232341498782,"chan":5,"rfch":0,"freq":433.575000,"stat":
1,"modu":"LORA","datr":"SF12BW125","codr":"4/5","lsnr":3.5,"rssi":-94,"size":17,"data":"gJq2/wCAgA0DMQAqvWzzrCg="}]}
INFO: [down] PULL_ACK received in 0 ms
INFO: [down] PULL_RESP received  - token[220:104] :)

JSON down: {"txpk":{"modu":"LORA","data":"oJq2/wCgfw0DzPqCog+TW3s=","freq":433.175,"prea":8,"tmst":2.187694652E9,"rfch":
0,"imme":false,"codr":"4/5","size":17,"ncrc":false,"datr":"SF12BW125","ipol":false,"powe":27}}
##### CLASS A here #####
**********JSON down Payload**********
00->a0 01->9a 02->b6 03->ff 04->00 05->a0 06->7f 07->0d 08->03 09->cc 10->fa 11->82 12->a2 13->af 14->93 15->5b
src/jitqueue.c:233:jit_enqueue(): ERROR: Packet REJECTED, already too late to send it (current=2235202206, packet=2187694652, type=0)
ERROR: Packet REJECTED (jit error=1)
INFO: [down] PULL_RESP received  - token[220:104] :)
```

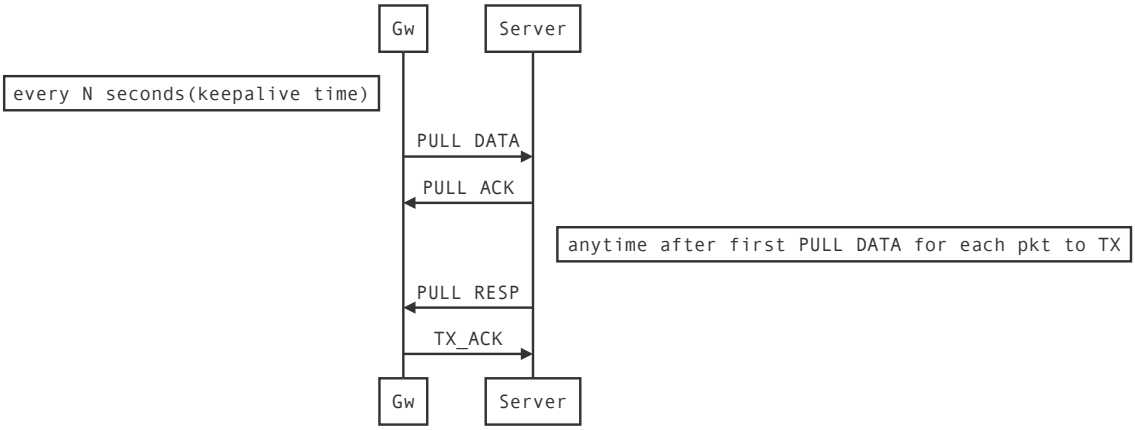
可以看到上行的上传时间是`22327335058`，然后收到了服务器的下行，时间戳是`2187694652`。按时序分析一下是。



最后一个服务器下行对应的不是`add=00FFB69A, fcnt=3456`，而对应的`fcnt=3455`，这个包在几十秒前就发送了，现在才收到对应下行。因此总是产生“too late”拒绝。

因此，真正的问题是：**上行发送过多过快，服务器下行回复太慢**

那么对于这个问题的原因，是因为keep alive机制导致的。根据网关-服务器下行通信协议



对比代码发现，代码实现与协议不符

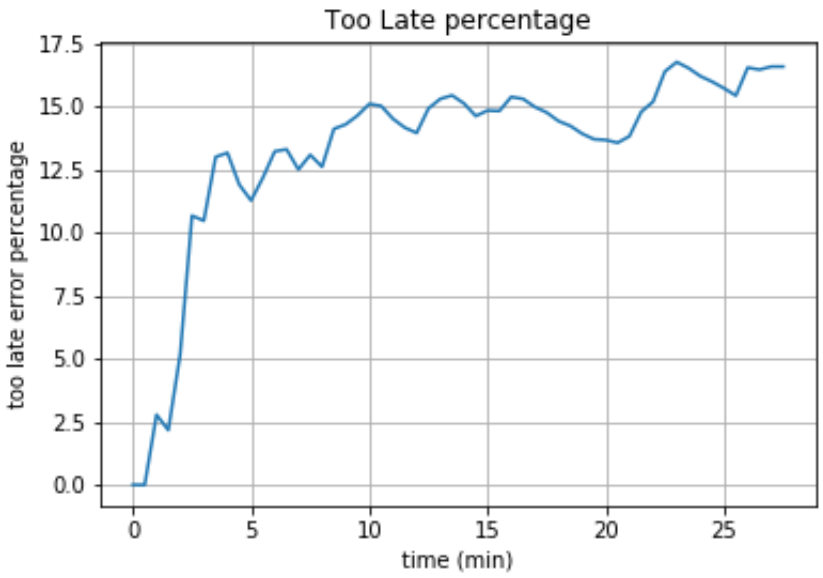
- `PULL DATA`本意指上行数据，下行现场中的`PULL DATA`实质只是2个字节的`PULL REQ`。

```
//下行代码
void thread_down(void){
    变量初始化
```

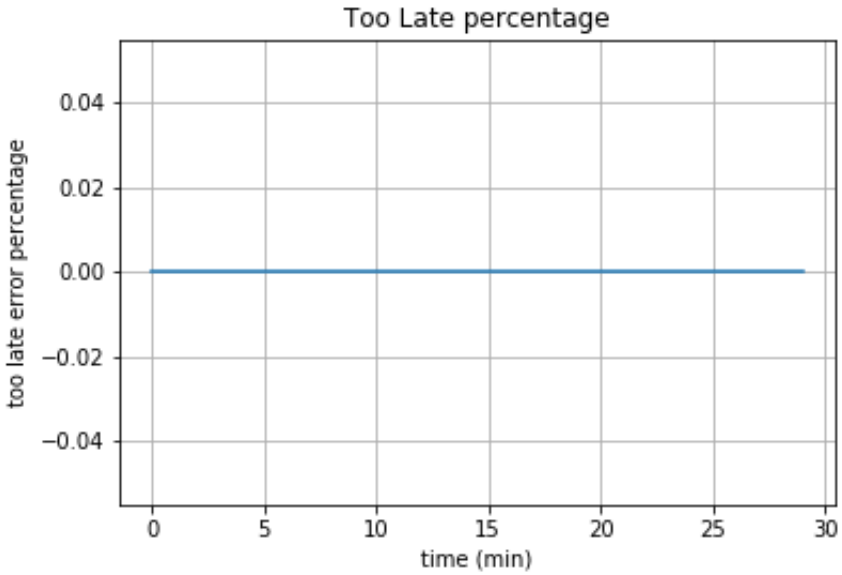
```
    设置socket
    填充buff_req
    while(1) {
        填充buff_req里的token
        把buff_req发给服务器，记录send_time
        recv_time = send_time;
        while(recv_time - send_time < keepalive_time){
            更新recv_time
            msg_len = recv(buff_down); //从服务器端接受buff_down
            if(msg_len == -1)
                continue;
            else
                根据buff_down处理
            // if(msg_len > 0) break;
        }
    }
}
```

- `while(1)`循环中，每次循环发送一个buff_req，也就是`PULL_REQ`，不是`PULL_DATA`。
- 进入内层循环后，在`keepalive time`内等待服务器下行。但处理完成后，大概率依旧在此循环内，**并且收不到新下行**，所以我猜测，每个服务器下行都要由一个`PULL_REQ`触发，因此做了两组测试，测试`too late`拒绝率
    1. 测试1：代码中不包含第17行，每`keepalive time`只发送一次`PULL_REQ`
    2. 测试2：代码中包含第17行，收到下行后跳出循环，发送下一个`PULL_REQ`

测试结果1:



测试结果2:



结论

1. 网关-服务器下行通信代码实现与协议不符合
2. 每个`PULL_REQ`只能触发一次`recv`接收（原因不知道是因为服务器还是socket）