# EAP指标统计不准问题的解决方案

**背景：**

每隔30秒定时器超时统一由目前指标的累加值减去上次保存的值进行打点。另外，我们引入了因其他原因导致指标打点失败的情况，计算公式如下:

其他原因次数 = 请求次数 – 成功次数 – 已知原因次数

这个计算公式对于目前的机制来说是有问题的，原因是当用户发起请求的时候，如果认证失败了，是要等几秒钟才能收到AAA发给用户认证失败的回应。假如我们还没有收到回应，30秒定时器就超时了，那么这时候会将没有回应的请求都算成因其他原因而失败，但是等到这次打点之后，等下一次，回应来了，将会把之前算出其他原因失败的，全部重新当成已知原因再打一次点。这样就造成了重复打点。导致指标值持续增长。当然认证成功或者中间的认证流程也是有延迟的，只是延时很短，暂不考虑。延时的情况，现在只靠考虑认证失败（失败会重传）还有超时（包括网络不通超时和AC内部不通超时）。

也就是说，只要我们引入了其他原因的概念，都会造成这个问题。

**方案一：忽略掉因定时器超时打点到下一次时间段内的指标。**

这也就是说，如果我认证请求发出了，没有在这个带点时段内（假设30秒为一打点的时段）回我认证成功或者认证失败，我都算作other原因失败，并且在下一个时段内，再收到延迟到来的回应消息，我也不累加了。

我们考虑到边界条件的两种情况（收到延时的回应消息时，全部都在下一个打点时段，或者一半在前一个时段，一半在后一个时段），来例举一下这种方案的两个问题：

1. 假设延时的回应消息全部都在下一个时段。

这种情况bug很多， 假设 时段一failother == 时段二fail，并且时段二没有请求次数，那么就好说，直接忽略掉时段二的fail。

但是，如果 时段一failother < 时段二fail，说明时段二有新的请求并且有新的失败原因。那么你怎么知道时段一failother是对应了时段二的那些fail呢？such as失败原因有我们认证超时和失败，你怎么知道，是超时了还是失败了? 这个是很难搞的，基本上搞不定，只能根据经验去优先忽略掉那种经常被延时的指标（也就是对延时导致failother的指标进行优先级排序，谁经常导致failother就先忽略谁，依次搞下来，直到数值相等符合公式）。

1. 假设延时的回应消息，一半在当前时段，一半在下一个时段。

这种我跟第一种其实区别不大，因为我们只关注failother的那部分。

总结：方案一不仅需要忽略一部分指标的真实失败原因，还会错误统计一部分指标的真实失败原因。如果考虑到大量用户上线并且全部失败都延时的特殊情况，可能会导致失败原因全部是failother，也就是未知。那么统计的意义将大幅度下降。如果再考虑到精确地延时，不仅仅包括失败和超时等等，还包括流程中，如请求和认证成功恰好打到了相邻的两个时段会导致认证成功也会被计入因其他原因而失败。

结论：这种方案没有办法根本性地解决问题。

**方案二：指标打点延时到下一时段时，不统计failother，将请求次数挪到下一时段，等到认证结果返回再一起统计。**

这个方案也不行，因为如果下一时段也有request，根本就搞不清楚，失败原因是属于哪一个时段的，可能造成误统计。

结论：这个方案也没法玩。

通过方案一和方案二，我们发现有两个问题需要解决才能从根源上把指标统计准确。

1. 必须搞清楚某个用户req后是否收到了回应，收到了什么回应。无法分辨某个用户具体已经走过了哪些指标流程，就没法精准地打点。
2. 要解决延时导致的failother打点问题，就必须将req打点后延，等到认证结果确定才打点。也就是根据第一点提出的，建立完整的一对一关系，并且将打点放在一对一关系确认之后。

综上，我提出方案三。

方案三：

first：在用户EAP表项中增加一个结构体变量，记录用户的打点状态，判断用户是否完整地达成了一对一状态（请求得到对应的回应）。

second：并在打点请求时创建30秒定时器。

1. 若定时器未超时，用户已经是一对一状态，则关闭定时器。
2. 若定时器超时，则将用户req的对应的指标记为failother。30秒已经足够让用户做完任何需要延时的操作了，所以必定是其他原因。（这点可以再确认下。）

third：只有一对一状态的指标数值才计入打点的全局变量中。

结论：方案三在数值上可以精确地打点，完美地符合计算公式。但是request打点时间会有最多30秒的延迟。另外造成AC性能很小程度的下降。