part1 DPDK APIs

meter

1. ```
   int rte_meter_srtcm_profile_config(struct rte_meter_srtcm_profile *p, struct rte_meter_srtcm_params *params);
   ```

   初始化srTCM的static数据，用 `params` 初始化 `p`。设置srTCM的CBS和EBS。并根据param的CIR设置srTCM更新`tc`和`te`的时间和数量。

2. ```
   int rte_meter_srtcm_config(struct rte_meter_srtcm *m, struct rte_meter_srtcm_profile *p);
   ```

   初始化srTCM的runtime数据，用已经初始化的`p`初始化`m`。设置`tc`，`te`为满，并记录初始时间。

3. ```
   static inline enum rte_color rte_meter_srtcm_color_blind_check(struct rte_meter_srtcm *m, struct rte_meter_srtcm_profile *p, uint64_t time, uint32_t pkt_len);
   ```

   根据profile和runtime data，得到当前包的颜色。传入的time会与上一次time相减，根据时间差和CRI增加tc和te的数目，再根据pkt_len得到颜色。

sched

1. ```
   int rte_red_config_init(struct rte_red_config *red_cfg, const uint16_t wq_log2, const uint16_t min_th, const uint16_t max_th, const uint16_t maxp_inv);
   ```

   初始化RED的config数据。`min_th`和`max_th`表示两个阀值。`maxp_inv`表示最大drop rate 的倒数。

2. ```
   int rte_red_rt_data_init(struct rte_red *red);
   ```

   初始化RED runtime数据。初始化avg，count（连续accept个数），q_time（距离队列为空的时间）为0。

3. ```
   void rte_red_mark_queue_empty(struct rte_red *red, const uint64_t time);
   ```

   队列为空时，让runtime 数据记录当前时间。

4.
```
int rte_red_enqueue(const struct rte_red_config *red_cfg, struct
rte_red *red, const unsigned q, const uint64_t time);
```

根据config 和runtime数据，以及当前队列大小和时间 通过以下公式，给出是否accept包。

$avg[i] = (1 - w_q) \times avg[i-1] + w_q \times q[i] \quad (q \neq 0)$

$avg[i] = avg[i-1] \times (1 - w_q)^m \quad (q = 0)$

$p_b = 0 \quad (avg < min_{th})$

$p_b = max_p(\frac{avg - min_{th}}{max_{th} - min_{th}}) \quad (min_{th} \leq avg < max_{th})$

$p_b = 1 \quad (avg \geq max_{th})$

$p_a = \frac{p_b}{(2 - count \times p_b)}$

part2 parameter deduce

1. 对每个flow设置一个srTCM的config和runtime数据，每个flow每个颜色设置一个REDconfig数据，每个flow一个RED runtime数据。
2. 每个flow记录模拟queue的长度，当不drop时长度加一。当time改变时，说明一个period（1000000ns）已过，清空每个queue。
3. flow0的srTCM: 环境最大send流量为 1000(一个burst平均包数)*640(一个包平均大小)byte * 8(bit)/4(4个流)/1000000ns=1.28GBit/s。这正好是flow0要求的流量，所以flow0不能丢包。*所以CIR为flow0输入的平均流量，设为为1.28G/8=160000000byte/s。CBS为burst的大小，这里平均为640*1000byte。EBS用来处理超出平均burst的包，设为CBS一半。
4. flow0的RED: max_th为队列的最大长度，因为平均一个flow的一个burst有1000/4=250个包，设为270。为了让flow0不丢包，min_th设置得靠近max_th，并且maxp_inv设置得较大（max_p较小）。wq_log2设置的大一点，这样计算avg时，会更多考虑历史queue大小，可以让avg变得更平稳，防止轻易超过270平均值丢包。
5. flow1的流量要为flow0的一半，所以将CIR,CBS,EBS都调小一半。表示标为黄色和红色的情况下的流量都为原来一半。同时减小max_th，增大wq_log2，让avg更容易超过max_th。减小min_th，减小maxp_inv(增大 maxp)，让提前drop的包更多。不断调节得到流量约为总量一半。
6. flow2，flow3同理可得。