/linux-3.13.1/net/sched/sch_api.c

```
extern struct pid *find_vpid(int);
extern struct task_struct *find_task_by_pid(int);
```

s64 get_current_dilated_time(struct task_struct *task)

```
{
	s64 temp_past_physical_time;
	struct timeval tv;
	s64 virt_start_time;
	s64 freeze_time;
	s64 task_past_physical_time;
	s64 past_virtual_time;
	int dilation_factor;
	s64 now;

	do_gettimeofday(&tv);
	now = timeval_to_ns(&tv);

	if(task == NULL)
		return now;

	virt_start_time = task->virt_start_time;
	freeze_time = task->freeze_time;
	task_past_physical_time = task->past_physical_time;
	past_virtual_time = task->past_virtual_time;
	dilation_factor = task->dilation_factor;

	if(virt_start_time > 0){

		s32 rem;
		s64 real_running_time;
		s64 dilated_running_time;
		real_running_time = now - virt_start_time;
		if (freeze_time != 0)
			temp_past_physical_time = task_past_physical_time + (now - freeze_time);
		else
			temp_past_physical_time = task_past_physical_time;

		if (dilation_factor > 0) {
			dilated_running_time = div_s64_rem( (real_running_time -
temp_past_physical_time)*1000 ,dilation_factor,&rem) + past_virtual_time;

			now = dilated_running_time + virt_start_time;
```

```
            }
            else if (dilation_factor < 0) {
                dilated_running_time = div_s64_rem( (real_running_time - temp_past_physical_time)*(dilation_factor*-1),1000,&rem) + past_virtual_time;
                now =  dilated_running_time + virt_start_time;

            }
            else {
                dilated_running_time = (real_running_time - temp_past_physical_time) + past_virtual_time;
                now = dilated_running_time + virt_start_time;
            }

        }

        return now;

}

EXPORT_SYMBOL(get_current_dilated_time);

struct task_struct * get_task_struct_from_qdisc(struct Qdisc * qdisc) {

        struct netem_sched_data *q = qdisc_priv(qdisc);
        struct qdisc_watchdog * wd = &q->watchdog;
        struct pid *owner_pid;
        struct task_struct * result;

        if(wd->owner_pid > 0) {

          owner_pid = wd->owner_pid;
          result = pid_task(owner_pid, PIDTYPE_PID);
                return result;

       }

        if(qdisc != NULL) {
            if(qdisc->dev_queue != NULL) {
              if(qdisc->dev_queue->dev != NULL) {
                  read_lock(&dev_base_lock);
                  if(qdisc->dev_queue->dev->owner_pid > 0) {
                owner_pid = qdisc->dev_queue->dev->owner_pid;
                read_unlock(&dev_base_lock);
                result = pid_task(owner_pid, PIDTYPE_PID);
```

```
                    if(result != NULL) {
            wd->owner_pid = owner_pid;
               }
               return result;
                  }
               read_unlock(&dev_base_lock);
            }
         }
      }

      return NULL;
}
```

static enum hrtimer_restart qdisc_watchdog_dilated(struct hrtimer *timer)

```
{

         struct netem_skb_cb * cb = NULL;
         struct task_struct * pkt_owner = NULL;
         s64 current_dilated_time = 0;
         struct qdisc_watchdog *wd = container_of(timer, struct qdisc_watchdog, timer_dilated);

         if(wd->skb == NULL)
            cb = NULL;
         else
            cb = netem_skb_cb(wd->skb);

         pkt_owner = get_task_struct_from_qdisc(wd->qdisc);

         if (pkt_owner == NULL || cb == NULL)
         {

          if(pkt_owner == NULL)
          printk(KERN_INFO "Netem: pkt_owner is null. Maybe sending early. Pid = %d\n",
wd->owner_pid);

          if(cb == NULL)
          printk(KERN_INFO "Netem: cb is null. Maybe sending early. Pid = %d\n",
wd->owner_pid);

          qdisc_unthrottled(wd->qdisc);
          __netif_schedule(qdisc_root(wd->qdisc));
         return HRTIMER_NORESTART;
         }
```

```
        current_dilated_time = get_current_dilated_time(pkt_owner);

        if (PSCHED_NS2TICKS(current_dilated_time) >= cb->time_to_send)
        {
         qdisc_unthrottled(wd->qdisc);
         __netif_schedule(qdisc_root(wd->qdisc));

        return HRTIMER_NORESTART;
         }
         else
         {
        hrtimer_forward_now(timer,ns_to_ktime(100000));
        return HRTIMER_RESTART;
         }
}
```

EXPORT_SYMBOL(get_task_struct_from_qdisc);

void qdisc_watchdog_schedule_ns_dilated(struct qdisc_watchdog *wd, u64 expires)

```
{
    if (test_bit(__QDISC_STATE_DEACTIVATED,
                 &qdisc_root_sleeping(wd->qdisc)->state))
        return;

    qdisc_throttled(wd->qdisc);

    hrtimer_start(&wd->timer_dilated,
                  ns_to_ktime(1000000),
                  HRTIMER_MODE_REL);

}
```

EXPORT_SYMBOL(qdisc_watchdog_schedule_ns_dilated);

/linux-3.13.1/net/sched/sch_netem.c

~~struct netem_sched_data~~
~~struct netem_skb_cb~~
~~static inline struct netem_skb_cb *netem_skb_cb(struct sk_buff *skb)~~

…time_to_send…

…tstamp_save…

/linux-3.13.1/net/packet/af_packet.c

extern struct pid * find_vpid(int);
extern s64 get_current_dilated_time(struct task_struct *);

/linux-3.13.1/include/net/pkt_sched.h

struct netem_skb_cb
static inline struct netem_skb_cb *netem_skb_cb(struct sk_buff *skb)
struct qdisc_watchdog(MODIFIED)
struct netem_sched_data

static inline void qdisc_watchdog_schedule_dilated(struct qdisc_watchdog *wd, psched_time_t expires)