****
9810 Tue Jan 15 10:39:02 2019
new/usr/src/lib/libfakekernel/common/taskq.c
10098 task_alloc() in libfakekernel gets KM_NOSLEEP test wrong
****
_____***unchanged_portion_omitted_***

```
 93 static taskq_ent_t *
 94 task_alloc(taskq_t *tq, int tqflags)
 95 {
 96         taskq_ent_t *t;
 97         int rv;

 99 again:  if ((t = tq->tq_freelist) != NULL && tq->tq_nalloc >= tq->tq_minalloc) {
100                 tq->tq_freelist = t->tqent_next;
101         } else {
102                 if (tq->tq_nalloc >= tq->tq_maxalloc) {
103                         if (tqflags & KM_NOSLEEP)
103                         if (!(tqflags & KM_SLEEP))
104                                 return (NULL);

106                         /*
107                          * We don't want to exceed tq_maxalloc, but we can't
108                          * wait for other tasks to complete (and thus free up
109                          * task structures) without risking deadlock with
110                          * the caller.  So, we just delay for one second
111                          * to throttle the allocation rate. If we have tasks
112                          * complete before one second timeout expires then
113                          * taskq_ent_free will signal us and we will
114                          * immediately retry the allocation.
115                          */
116                         tq->tq_maxalloc_wait++;
117                         rv = cv_timedwait(&tq->tq_maxalloc_cv,
118                             &tq->tq_lock, ddi_get_lbolt() + hz);
119                         tq->tq_maxalloc_wait--;
120                         if (rv > 0)
121                                 goto again;             /* signaled */
122                 }
123                 mutex_exit(&tq->tq_lock);

125                 t = kmem_alloc(sizeof (taskq_ent_t), tqflags);

127                 mutex_enter(&tq->tq_lock);
128                 if (t != NULL)
129                         tq->tq_nalloc++;
130         }
131         return (t);
132 }
```

_____***unchanged_portion_omitted_***