IEEE Standards Interpretations for IEEE Std 1003.1c™-1995 IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) - System Application Program Interface (API) Amendment 2: Threads Extension (C Language)

Interpretation Request #19
Topic: sched_setparam **Relevant Clauses:** 13.3.3.2, page 114 D10, lines 30-36 Clause 13.3.3.2, page 114 D10, lines 42-48

According to the new rules for scheduling, the sched_setparam() and sched_setscheduler() are not useless in the presence of multithreaded applications. The only effect these functions have is on the child process [from fork()] of the target process. An implementation may cause something to happen to the process scope threads, but that's all. This provides a huge hole for system administrators. These functions have a process parameter. This means they were intended so that a process could control the policy and priority of another process.

If that wasn't the case, there wouldn't have been a pid as a parameter. Up until now a system administrator could control a process if it was getting too much or too little time on the system. Runaway processes with high priority could be handled. With the new behavior, there is no way a sysadmin (or any process) can control the behavior of a multithreaded process. If some event happens requiring a change in the policy/priority of the MT process, only the process itself can do it. The worst part is that if one thread in the process goes out of control while a high priority SCHED_FIFO, no one can control it. A sysadmin cannot do anything to lower that thread's priority (thread IDs are not guaranteed to be known outside of the system). Is this the actual intended behavior?

Interpretation Response
This is a duplicate. See Interpretation #3, part 7.

Rationale for Interpretation: None.