

Case study: Mars Pathfinder

- The Mars Pathfinder successfully landed on on the martian surface on July 4th, 1997, deploying the Sojourner rover and sending picture and data back to Earth
- A few days after landing, the Pathfinder began experiencing total system resets, each resulting in losses of data.
- What follows is a short description of what happened, based on a account put together by Mike Jones of Microsoft Research
 - more details can be found on Jones' web site at http://research.microsoft.com/~mbj/Mars_Pathfinder/

Pathfinder: the main characters.

- The o.s. chosen for the mars Pathfinder mission was VxWorks
 - VxWorks is a commercial real-time embedded systems kernel made by Wind River Systems
 - it provides preemptive priority scheduling of threads
- Pathfinder contained an "information bus", which you can think of as a shared memory area
 - used for passing information between different components of the spacecraft.
 - Access to the bus was synchronized with mutual exclusion locks (mutexes)

Pathfinder: the villains

- A *bus management* task ran frequently with *high priority* to move certain kinds of data in and out of the information bus.
- a *communications task* that ran with *medium priority*
- The *meteorological data* gathering task ran as an infrequent, *low priority* thread
 - When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex

Pathfinder: the crime scene

- If an interrupt caused the information bus thread to be scheduled while the mutex was held by the meteo thread, and if the information bus thread then attempted to acquire this same mutex in order to retrieve published data, this would cause it to block on the mutex
 - This scenario is a classic case of priority inversion and it went undetected during design&testing

Pathfinder: the crime

- Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running.
- After some time had passed, a watchdog timer would go off, notice that the information bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset.

Pathfinder: the investigation

- After the failure, JPL engineers spent hours and hours running the system on the exact spacecraft replica in their lab with tracing turned on, attempting to replicate the precise conditions under which they believed that the reset occurred.
- Early in the morning, after all but one engineer had gone home, the engineer finally reproduced a system reset on the replica. Analysis of the trace revealed the priority inversion.

Pathfinder: the solution

- When created, a VxWorks mutex object accepts a boolean parameter that indicates whether priority inheritance should be performed by the mutex
 - unfortunately the mutex in question had been initialized with the parameter off
 - had it been on, the low-priority meteorological thread would have inherited the priority of the high-priority data bus thread blocked on it while it held the mutex, causing it be scheduled with higher priority than the medium-priority communications task, thus preventing the priority inversion.

Pathfinder: lessons

- Debugging process synchronization problem is hard
 - detailed traces of actual system behavior enabled the faulty execution sequence to be captured and identified.
 - leaving the "debugging" facilities in the final system saved the day
- the engineer's initial analysis that "the data bus task executes very frequently and is time-critical -- we shouldn't spend the extra time in it to perform priority inheritance" was wrong
 - JPL engineers later confessed that one or two system resets had occurred in their months of pre-flight testing. They had never been reproducible or explainable, and so the engineers, in a very human-nature response of denial, decided that they probably weren't important, using the rationale "it was probably caused by a hardware glitch".