DSN 2016 Presentation Preparation

songjiguo

June 22, 2016

Contents

1 Motivations and Challenges **1**
1.1 Page 4-5 1
1.2 Page 6-7 1
1.3 Page 8 2
1.4 Page 9-10 2
1.5 Page 19-20 3
1.6 Page 21-25 3

1 Motivations and Challenges

1.1 Page 4-5

on April 28th the Japan Aerospace Agency declared the satellite Hitmomi, on which it had spent $286 million USD, lost. Not to mention possible 10 years' worth of research lost as well. The reason behind this is still under investigation, but the evidence shows that the system experienced glitches when it passed through a region with an increased flux of energetic particles, which leads to a series of cascading failures."

1.2 Page 6-7

- Uncontrolled vehicle acceleration
  - "a bit-flip there, will have the effect of killing one of the tasks" – Michael Barr testimony, Bookout v Toyota
  - 831 Unintended Accelerations (UAs) in Camrys
  - the acceleration control task killed = UA

  - Control CPU software: 280000 LOC
  - Hardware: V850E1 – 32bit RISC
  - OS: OSEK-compliant OS with 24 RT tasks, Mostly C
  - replicated 30 second UA due to bit corruption of scheduler data-structures
  - Electronic Throttle Control System (ETCS)
  - resulted in settlement to avoid punitive damages
  - jury: toyota had acted with "reckless disregard" – NYTimes (Toyota Agrees to Settlement in Fatal Acceleration Crash – Oct 25, 2013)
  - "Oklahoma case is extraordinary because it is the first to try the electronics and software defects that plaintiffs say existed in the Toyota Models" – NYTimes
  - multiple on-going cases

- NASA has conjectured that transient faults could have caused the uncontrolled acceleration in their cars a few years ago, leading to a billion dollar lawsuit

1.3 Page 8

Transient faults, also called Single Event Upset (SEU. The radiation-induced error caused by a transient fault is called soft error. (Alpha particles from packaging material and high-energy ($>$ 1 MeV) neutrons from cosmic radiation can induce soft errors)

1.4 Page 9-10

Unfortunately, there are signs that this situation will only get worse with time. On one hand, the embedded system architectures are getting more and more complex. For example, compared to the car in 70's, a modern car can provide many more functionalities with up to 100 ECUs and the complex software of 100 million lines of code, which make the system dependability more challenging. On the other hand, while the semiconductor manufacturing processes keep descreasing the feature size, it also increases their susceptibility to the radiation faults. While getting smaller, faster, more energy efficient chips, it requires that we need address transient faults

1.5 Page 19-20

Our previous work C3 investigated how to predictably recover from the system-level fault in a component-based OS and here we give a very brief and high-level review about how C3 works.

C3 is built on the top of Composite in which the system services such as the scheduler, memory manage and file system are all implemented as the user-level components and each component has its own protection domain + kernel does not have policy

1.6 Page 21-25

C3 adds some intelligent code at the interface between components to track the system state and help the recovery, as shown in the blue blocks. For example, here we have 3 components – application, lock and scheduler component and the black solid lock and circle shape represent the lock and thread data structure respectively. The application component could invoke the lock component to take or release a lock. The lock component could also invoke the scheduler component to block or wake up threads. The scheduler component could invoke the kernel to dispatch the threads.

The fault that occurs in the lock component can corrupt the data structure in it. The faulty component will be first micro-rebooted and then during the recovery, the tracking information is used with the component operations (or component interface functions) to reconstruct the consistent state in the faulty component.