ECEN5623, Real-Time Systems   Exercise #6 – Learning from Failure, Bettering Proposals

Vidur Sarin (vidur.sarin@colorado.edu), Harsimran Singh Bindra (harsimran.bindra@colorado.edu), Arundhathi Swami (arundhathi.swami@colorado.edu)

Q1: Research, identify and briefly describe the 3 worst real-time mission critical designs errors (and/or implementation errors) of all time [some candidates are Three Mile Island, Mars Observer, Ariane 5-501, Cluster spacecraft, Mars Climate Orbiter, ATT 4ESS Upgrade, Therac-25, Toyota ABS Software]. Note that Apollo 11 and Mars Pathfinder had anomalies while on mission, but quick thinking and good design helped save those missions]. State why the systems failed in terms of real-time requirements (deterministic or predictable response requirements) and if a real-time design error can be considered the root cause.
ANS:

1. Crash of AT & T network, 1990

Effect: During this period, AT&T supported about 70% of the country's long- distance phone traffic based off of its 114 computer operated electronic switches. On January 15th 1990, AT & T's network experienced failures at multiple switching stations throughout the country. This scenario plagued the network for close to 9 hours during which AT & T lost 50% calls (50 Million calls were blocked) that tried routing through its network. AT & T lost about $60 Million in unconnected calls and this statistic doesn't include any losses incurred by businesses that relied on AT& Ts network to support its calls.

Cause: The network of 114 computer-operated electronic switches are each capable of handling 700000 calls an hour and are linked via a cascading network (common channel signaling system-7). Each of these switches take an average of **4-6 seconds** to route a call through the network from source to destination. When a switch received a call, it scanned a list of 14 different possible routes to complete the call while simultaneously passing the calling number to a parallel signaling network to check for alternative routes to the destination. On finding the destination switch available, a reservation was made at the destination switch to pass the call through to its local network. If the destination was busy, the caller was notified through the original switch and the line freed.
The problem originated in the switch stationed in New York City which performed a routine self-test. This self-test indicated that the switch was nearing its loading limit and would be unable to accept any more incoming calls. As was the protocol, the station performed a maintenance reset that lasted 4 seconds and sent a message to all connected stations that it wouldn't take any more calls until further notice. The engineers called this kind of message a **congestion message.**
Previously these congestion messages updated the connected switches about the inactivity of the originator and when the originator was back online, it resent a message stating that it was now in service. In the updated software however, no such reconnection message was sent. If the connected switches observed messages from the previously inactive switch, they updated their tables to reflect the availability of the switch. Hence when the New York City switch returned online, it began to relay messages that had backed up in its server and the connected stations updated their tables to show that the NYC switch is back online.   However the switch received a second message for connection from the NYC server within 10 ms. Since this second message was received before the first message was processed, it was written over onto crucial communications information instead of being buffered due to a software bug. Software in the switch immediately detected the over write and ran its own 4 second reset sequence.
The entire sequence then repeated with the second server and all its connected servers thus exponentially affecting all servers in the network.
It took over 9 hours and reduced network traffic to stabilize the system against the reoccurrence of this sequence of events.

Real Time Critical Design Error: The software bug that resulted in the reset sequence being initiated due to overwrites caused by sparsely spaced messages was traced back to a software upgrade intended to make the system process congestion and other notification messages faster and with higher priority. Since this software upgrade was conducted on all 114 switches, each of them reacted in the same way leading to a cascading failure of the entire country wide network. The entire software bug can be traced to a break statement within the block of code that processes an incoming message. If the incoming message was form a switch that was out of service, it checked the ring write buffer. If the ring write buffer was empty as well, it updated the status of the switch to in service or online to the status table. If it failed both conditions, it caused a break effectively exiting the incoming message case. When the messages arrived that were closely times, it failed the second or ring buffer empty clause and broke out of the incoming messages case statement and proceeded to "do optional parameter work". This ended up in over write of crucial communications data and caused a system reset.

Reason for System Failure in terms of real time requirements: The software was upgraded to improve the real time response to crucial systems messages such as congestion messages. Since the response time improved, the gap of 4 secs required to process a call or incoming message on a switch was overridden. This discrepancy lead to hitherto unseen responses to incoming calls.

Pivotal Cause of Failure: The pivotal call can be classified as a soft real-time error that provided a uniform but unpredictable error due to a single line software bug.

References:
Ref1
Ref2

2. Therac 25, 1985-87

Effect: Therac-25 was a radiation therapy machine (medical linear accelerator) used to provide radiation therapy to cancer patients suffering from malignant tumors. As the name suggests, these machines accelerated electrons that created energy beams that could potentially be used to shrink and ultimately destroy tumors. For shallow tissue penetration electrons were used and to access deeper tissues, beams were converted to x-rays. To aid this segregation, the machine had two principal modes of operation – low energy mode to provide electron beam rays and a high energy mode that required placement of a thick metal slab between the patient and the accelerator to convert the beams into x-rays before reaching the patient. However due to incorrect error handling and overlapping of both modes of operation, the machine emitted about 100x more than the safe level of radiation to the patients without the knowledge of the operating technicians. This resulted in the patients suffering from untreatable radiation burns which they succumbed to.

Cause: There are several reasons responsible for the occurrence of this grave oversight. Two of the main reasons can be identified as bad software design and bad error message display for debug purposes and use purposes. The specific sequence of events that caused this misadministration of radiation is as follows:

1. The technician pressed the key **x** – for xray radiation instead of **e** – for electron beam radiation.
2. This was followed by **enter** which indicated to the machine that the treatment is ready to start
3. The computer responded with a **beam ready signal –** which indicated that the machine is about to start electron beam treatment
4. The technician followed this with a **b** which is the signal the machine waits on to start delivering the radiation to the patient

5. The computer then output a generic error message which the technician chose to override and repeated the process of entry from the beginning
6. This entire sequence occurred within 8 secs which the machine was not tested for.

While the technician was resetting the machine, with every error message the machine was spitting our x rays towards the patient in a very concentrated area. Since the machine was unable to comprehend whether the required settings were for electron beam or x ray, it chose to retract the metal slab. The patient was thus directly subjected to a very high concentration of x-ray radiation.
The machine was not set up to recognize discrepancies in entry of commands and did not particularly help the technician comprehend what was occurring inside the chamber either. More importantly there was no hardware in place to ensure the presence of the metal slab for instances of xray radiation or even to halt operation in case of error messages.

Real Time Critical Design Error: Majority of the mistakes in this case were due to inadequate testing of the system ad improper deadlines for the system. The system was confused between its softwares decisions and hardwares behavior due to the fact that the instruction sequence changed by the technician occurred within **8 seconds.** Within the software design itself, the communication between individual threads of operation were not well organized or synchronized.

Pivotal Cause of Failure: This new version of the Therac machine – 25 was entirely software controlled with no hardware involved in any safety contingency schemes. Error logging was highly unspecific and inadequate.

References:
Ref1
Ref2

3. ABS Recall Toyota, 2010
Effect: ABS is Anti-Lock Braking System. This system is lately being used for stability control and a crude aid to traction to help drivers exert better control while turning corners or braking and steering simultaneously. In February 2010, Toyota received complaints from 102 drivers in USA and 14 from drivers in Japan about possible problems in the braking mechanism of their Prius model. One incident in July 2019 ended with a Prius crashing into the car ahead and injuring two people. When the potential ramifications of this occurrence were considered, Toyota announced a voluntary recall of close to 133000 vehicles in the US and 52,000 in Europe were given software update to combat the issue. Toyota was also charged criminally with a civil class action lawsuit for the way it dealt with the recall of vehicles. All of this was in addition to unaccountable losses incurred by Toyota in potential sales.

Cause: The basic mechanism is that an ABS sensor is attached to each wheel that detects when the wheel is being stopped (i.e locked in terms of braking control). At this instance the retardation is much lower than the generic car movement and the car cannot be steered, thus diminishing the drivers control over the car. However, if the brake pressure being - applied to the wheel axels through a hydraulic fluid that is pumped through the system as a result of the brake pedal being pressed on – is released momentarily, the brakes are freed and the car can be revolved again.
The ABS in normal operation, engages and disengages rapidly as the control system senses and reacts to tire slippage. This prevents the car going into a brake-induced skid and enabling the driver to continue to steer and maintain control over the car.
However, the new ABS software updates made on the Prius introduced the short delay in regenerative braking when hitting a bump, resulting in an increased shopping distance. This anomaly was not immediately noticed or intuitively sensed by drivers, thus causing the loss in control and consequent crashes.

Real Time Critical Design Error: The critical problem in the software update to the electronically controlled braking system introduced an unintended delay when switching between the mechanical hydraulic brakes and the electronic brakes. This delay in switching was the catalyst in the time lag the drivers noticed before the brake starts to function when they are driving over a bump or a slippery surface. This delay makes the brake system respond non-deterministically during every application. The fix to this issue to keep pressing the brake pedal until the delay is overcome and the electronic brake system kicks in.

Pivotal Cause of Failure: Glitch in the brake that causes hardware to introduce a delay in the sticking of brakes.

References:
Ref1
Ref2
Ref3

➔ **Ranking**

1. **Therac 25, 1985-87**
   The disaster was responsible for the death of 6 terminally-ill people in a gruesome manner.
2. **ABS Recall Toyota, 2010**
   Toyota updated the software of nearly 200,000 vehicles and the potential for accidents was manifold had the recall not been executed. The financial ramifications for Toyota in terms of providing free software upgrades as well as replacing recalls in addition to the payments for settling law suits were grave. In addition to that, the loss in terms of trust from customers is unaccountable.
3. **Crash of AT&T Network, 1990**
   AT&T ended up facing huge financial losses and was directly responsible for any losses incurred by businesses that depended on its network as well.

Q2: The USS Yorktown is reported to have had a real-time interactive system failure after an upgrade to use a distributed Windows NT mission operations support system. Papers on the incident that left the USS Yorktown disabled at sea have been uploaded to D2L for your review, but please also do your own research on the topic.

a) [4 pts] Provide a summary of key findings by Gregory Slabodkin as reported in GCN.
b) [4 pts] Can you find any papers written by other authors that disagree with the key findings of Gregory Slabodkin as reported in GCN? If so, what are the alternate key findings?
c) [4 pts] Based on your understanding, describe what you believe to be the root cause of the fatal and near fatal accidents involving this machine and whether the root cause at all involves the operator interface.
d) [4 pts] Do you believe that upgrade of the Aegis systems to RedHawk Linux or another variety of real-time Linux would help reduce operational anomalies and defects compared to adaptation of Windows or use of a traditional RTOS or Cyclic Executive? Please give at least 2 reasons why or why you would or would not recommend Linux.

ANS:

a) Some of the key findings by Gregory Slabodkin are as follows:

The Yorktown Smart Ship began running shipboard applications under Microsoft Windows NT so that fewer sailors would be needed to control key ship functions.

Some of the failures encountered on the Yorktown Smart Ship were:

- A systems failure when bad data was fed into its computers during maneuvers.
- A database overflow caused its propulsion system to fail
- Lost control of its propulsion system because its computers were unable to divide by 0 causing the database to overflow and crash all LAN consoles and miniature remote terminal units.

b) No, there are no alternate papers that disagree with Slabodkin's key findings. There are other papers by the same author but none that disagree. Some of the authors other papers are:

https://www.wired.com/1998/07/sunk-by-windows-nt/

There are however, some reasons why Windows NT may have been chosen:
- politics played in assigning of the contract.
- Windows NT has a better GUI that may have been a reason for choosing it over Unix.
- Bill Gates influence. Bill Gates nominated the Smart Ship program for the ComputerWorld/Smithsonian Awards Program.

c) The root cause of some of the errors are as follows:

- Bad data being fed into an application running on one of the 16 computers on board. The data contained a zero where is shouldn't have and when the software attempted to divide by 0, it crashed causing the ship to lose control of its propulsion system.

- A database overflow that caused the propulsion systems to fail.

- Apart from the above technical issues, I believe that an unbiased assessment of the 2 OS should have been done before porting onto the on-board systems. This would have led to a measured decision to adopt a particular OS over another.

d) Based on the context of the problem, the issue has nothing to do with hard/soft real time systems. This issue should have been solved by including a fault handler in the software. This could have been done by simply patching the program. However, there are other issues with using Windows NT over a hard-real time Linux distribution(RedHawk).

2 reasons why I'd suggest any scalable hard-real time OS are:

- Hard real-time systems will give priority to certain tasks that are more important and always have deterministic deadlines. This is critical in a military vessel where any failure can lead to catastrophe.

- More operation control over preemption and blocking/yielding hence leads to better scheduling and control on the system.

Q3: Form a team of 2, 3, 4, or 5 students and propose a final Real-Time prototype, experiment, or design exercise to do as a team. The proposal should either:

1) Provide a design and prototype or a real-time application with 2 or more services with deadlines,
2) Provide a design and prototype to address a current (contemporary) real-time application such as intelligent-transportation, UAV/UAS sense-and-avoid operations, interactive robotics, etc., or
3) Design and experiment to evaluate how well Linux on the DE1-SoC or Jetson provides predictable response for real-time applications with at least one interrupt driven sensor and an observable output (e.g. audio, video display, actuation of a device).

For option #1, you could for example design and prototype a camera system to track a bright object (laser target designator) in a room, implement it, test it, and describe your design and potential improvements. For option #2, identify any contemporary emergent real-time application and prototype a specific feature (e.g. lane departure and steering correction for intelligent transportation), design, implement a prototype with the DE1-SoC or Jetson, and describe a more complete real-time system design. For option #3, pick a test or simulated set of services (2 or more) which you can evaluate with Cheddar, with your own analysis, and then test with tracing and profiling to determine how well Linux really works for predictable response. You will be expected to write your own requirements. You will still need to provide a report which meets requirements outlined in Exercise #6 and make a presentation for your final exam, but the exact format should be tailored to your creative analysis, design and implementation.

A. [20 pts] Your Group should submit a proposal that outlines your exercise in real-time systems design and prototyping/testing with all group members clearly identified. This should include some research with citations (at least 3) or papers read, key methods to be used (from book references), and what you read and consulted.

B. [20 pts] Each individual should turn in a paragraph on their role in the project and an outline of what they intend to contribute.

ANS:
Please refer attached project proposal for the solution to this question.