The importance of this article is that it describes a method of providing software fault tolerance for deep-space missions. The article defines fault tolerance as a system being able to avoid system failure and maintain functionality when faults occur. The need for software fault tolerance is driven by the enhanced autonomy and onboard computational capabilities of future deep-space missions. Most of the previous space missions have utilized an Earth-bound control, where a significant number of failures could be handled by first putting the spacecraft in a safe mode while human controllers attempted to restore the spacecraft to its normal operational mode. In order to increase the success of deep-space missions, it is necessary to develop software based fault tolerances.

A fault is defined as a defect in a system. A fault can lead to an error occurring, once the incorrect program code is executed. Causes of faults during space missions include both equipment failure as well as environmental impacts, such as radiation, temperature extremes, or vibrations. Spacecraft that are operating in deep-space environments are subjected to cosmic ray and solar wind radiation which exposes them to protons, alpha particles, heavy ions, ultraviolet radiation, and X-rays. Fault tolerance is a necessity for deep-space missions, where immediate and continuous control from Earth is not available.

Although modern customer-off-the-shelf circuits are protected against long-term degradation and catastrophic effects caused by radiation, they are still susceptible to transient
faults such as single event upsets (SEUs) or multiple bit upsets (MBUs). An SEU changes the state of a single bit in the memory, and even though only bit has changed, failures can be created. Faults from an SEU can affect many different components, such as processor cores, memory units, memory controllers, communications networks, input/output processor nodes, and interconnection networks. This can result in corrupted data or crashes due to the execution of incorrect branches of code.

Adaptive fault tolerance allows for flexibility when dealing with SEUs. As an example, an SEU that causes a single bit flip in the initial phase of an image processing algorithm may not affect the outcome of the computation, and therefore would not require any action from the adaptive fault tolerance software. However, an SEU that resulted in the corruption of a critical data structure due to an illegal assignment to one of its components or a change of an instruction code or the corruption of an address computation would have devastating effects on the outcome of the computation.

Fault detectors can effectively be used to mask a fault by executing a redundant piece of code. The conceptual model presented in this article can be implemented as a software-based fault tolerance on future deep-space missions to compensate for the inability to offer immediate and continuous human support from Earth. Fault tolerance can be used onboard deep-space missions to help ensure their successes.
Citations:

