위성 S/W 개발을 위한 Java 기반의 Subsystem 시뮬레이터 구축

신현규, 최종욱, 이종인
한국항공우주연구원
e-mail : hkshin@kari.re.kr

Subsystem simulator using java for the satellite S/W development

Hyun-Kyu Shin, Jong-Wook Choi, Jong-In Lee
Korea Aerospace Research Institute

요 약

인공위성에 탑재되는 S/W 는 위성 내의 다양한 장치들과 유기적으로 통신하며 위성의 동작을 제어하고 임무를 수행한다. 따라서 이들 장치들과의 상호 작용이 충분히 테스트 되어야 하나, S/W 개발 과정에서 해당 장치들과 직접 연동되어 개발하기 어려운 경우가 대부분이다. 이에 위성에 탑재되는 다른 장치 및 탑재체의 기능과 역할을 모사할 수 있는 시뮬레이터가 필요성이 대두된다. 본 연구에서는 Java 를 이용한 시뮬레이터 개발 방안에 대하여 소개한다.

1. 서론

인공위성에 탑재되는 S/W 는 위성 전체의 동작을 제어하고, 부여된 임무를 수행하는데 있어서 없어서는 안 되는 필수적인 부분이다. 이러한 위성 S/W 는 위성 내에 탑재된 다른 장치(Subsystem)들과 유기적으로 통신하여 위성의 상태를 파악하고, 이에 따라 위성을 제어하게 된다. 위성 S/W 의 개발에 있어 다른 장치들과의 상호 작용이 충분히 테스트 되어야 하나, S/W 개발 과정에 있어서는 실제 장치에 대해 테스트를 수행하기 어려운 경우가 다수 존재하게 된다.

이에 위성에 탑재되는 다른 장치 및 탑재체의 기능과 역할을 대신할 수 있는 시뮬레이터가 요구되며, 이를 통해 보다 빠르게 S/W 를 개발하고 테스트하며, 실제 환경에서 발생할 수 있는 여러 문제점을 사전에 발견할 수 있다. 본 연구에서는 위성내의 전력 시스템을 관리하는 PCDU (Power Control and Distribution Unit) 에 대한 시뮬레이터 작성 방안에 대해서 소개한다.

2. 시뮬레이터 요구 사항 및 개발 환경

위성에 탑재되는 다양한 장치들은 태양 전지에 의해 생산된 전력을 이용하여 동작을 수행하게 된다. 이때 위성 내 다양한 장치들에게 전력을 분배하는 역할을 PCDU 가 수행하게 되는데, 위성 S/W 와 긴밀히 통신하며 위성 제어에 큰 역할을 맡게 된다. PCDU 는 위성 S/W 로부터 전달된 Command 를 수행하거나 현재의 전력 분배 상태를 위성 S/W 로 전달하게 된다.

PCDU 는 Command 가 전달되면 이에 대한 응답을 빠르게 위성 S/W 로 보내야 하며, 상태 정보의 전송 역시 제한된 시간 안에 완료해야 한다. 이는 위성 S/W 의 real-time 제약 조건에 기인하는 것으로, 실제 PCDU 는 이 제약 조건을 충족하도록 개발된다. PCDU 를 시뮬레이션하기 위한 시뮬레이터 역시 이러한 제약 조건을 충족해야 한다.

PCDU 시뮬레이터는 Linux 환경에서 Java 를 기반으로 작성되었으며, 빠른 통신을 위해 통신 부분은 Native Code 로 구현된 라이브러리를 사용하였다.

3. 시뮬레이터 모델링

시뮬레이터를 작성하기 위해 가장 먼저 수행한 작업은 PCDU 의 동작을 모델링하는 방법이다. 시뮬레이터는 실제 장치가 아니기에 내부 구현 방식과 관계없이 외부로 보여지는 동작에 대해서 작성하는 것이 효과적이기에 다음과 같은 동작을 수행하도록 구성하였다.

위성 S/W 로부터 전달된 Command 에 대해 지체 없이 응답을 보내며, 해당 Command 는 PCDU 가 올바르게 수행하였음을 가정하여 PCDU 의 상태 데이터를 변경시키도록 한다.

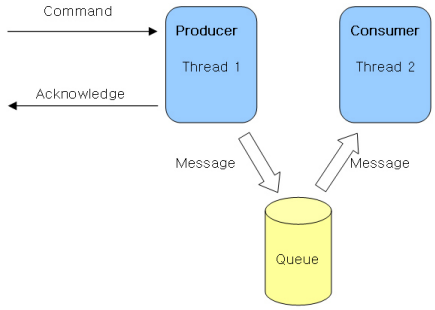
4. 시뮬레이터 작성 기법

위의 모델링 내용을 바탕으로 시간 제약 조건을 충족하기 위하여 다음과 같은 기법을 적용하였다.

첫째, 기능에 따라 Thread 를 분리하여, 통신을 담당하는 Thread, Command 를 해석하고 수행하는 Thread 로 구성하였다.

둘째, 이 Thread 들에 대하여 Producer/Consumer Pattern 을 적용하였다. Thread 에 의해 동작이 분리되더라도 전달된 Command 의 수행결과를 통신 Thread 가 기다리게 되면 요구된 시간 제약 조건을

충족하기 어려운 경우가 빈번히 발생하게 된다. 따라서 비동기적으로 작동이 가능하도록 Producer/Consumer Pattern 을 적용, Thread 간의 의존도를 낮추어 전체적인 반응속도를 향상시켰다. Thread 분리 및 패턴의 적용방법은 그림 1 과 같다.



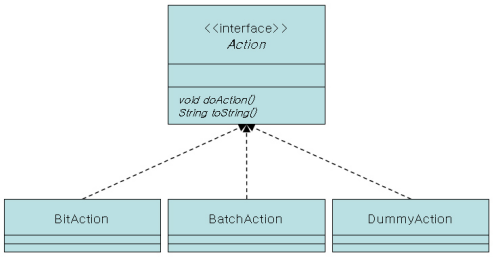
(그림 1) Thread 분리 및 패턴 적용

셋째, User Interface(UI) 에 대한 느린 Update 기법. 전달된 Command 는 PCDU 의 상태를 나타내는 테이블의 값을 변경하게 되며, 이는 UI 를 통해 사용자에게 알려지게 된다. Command 가 빠른 간격으로 PCDU 로 전달될 경우 UI 가 지속적으로 Update 되어야 하는데 이는 시간 제약 조건을 충족시키지 못하게 되는 주요 원인으로 작용함을 알 수 있었다. 따라서 사용자가 값을 확인하고 싶은 경우에 대해서 선택적으로 UI 를 Update 할 수 있는 기능을 추가하여, 정상적인 동작 상황에서는 UI Update 를 사용자가 필요로 하는 시점으로 유보하는 기법을 적용하였다.

넷째, 선택적 메시지 표시 기법. 시뮬레이터에 전달되는 명령이나 메시지에 대해 표시 레벨을 설정하여 표시되는 메시지의 양을 줄여 전체 소요 시간을 단축하였다. 동작을 검사하고 모니터링하는 경우는 전체 메시지 표시레벨을 선택하고, 정상적인 상태에서는 유효한 명령에 대한 메시지 만을 표시할 수 있도록 하였으며, 보다 상세한 내용이 필요한 경우 Command 에 의해 변경되어야 할 상태 데이터 및 현재의 상태 데이터 정보를 표시해주는 기능을 추가하였다.

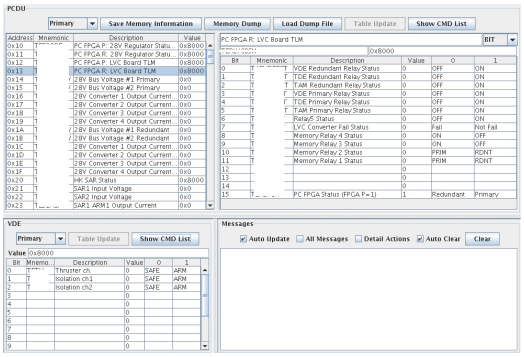
다섯째, 시뮬레이터의 메시지 표시 윈도우에 표시되는 메시지의 길이를 최소화하고, 자동으로 해당 내용을 삭제시킬 수 있는 기능을 추가하여 메시지 표시 과정에서 발생하는 Update overhead 를 감소시켰다.

시간 제약 조건의 충족 외에도 Command 처리의 효율성 및 향후 다른 위성의 개발에서 효과적으로 사용하기 위해 Command 를 소스 코드와 분리하여 별도의 파일로 유지, Text 형태와 XML 형식을 모두 지원하도록 하였다. Command 에 대한 PCDU 의 상태 데이터 변경 방식에는 Java AWT 의 Event 처리 방식을 응용하여 그림과 같이 Action 이라는 인터페이스를 BitAction, BatchAction, DummyAction 등으로 상세 구현하였다.



(그림 2) Action Hierarchy

5. 운용 결과



(그림 3) PCDU Simulator

그림 3 과 같이 시뮬레이터를 구현된 시뮬레이터를 이용하여 실제 개발환경에서 위성 S/W 를 이용하여 테스트 해본 결과 대부분의 경우에서 시간 제약 조건을 충족함을 확인해 볼 수 있었다. 위성 S/W 에서 전달하는 연속된 Command 에 대해 평균적으로 2msec 안에 응답을 확인할 수 있었다.

6. 결론

본 연구에서는 PCDU 의 동작을 모델링하고, 다양한 기법을 적용함으로써 시간 제약 조건을 충족시킬 수 있는 시뮬레이터를 개발, 운용하였다. 또한 Command 리스트 관리 및 처리에 대한 방법을 연구, 적용하여 Command 의 변경이나, 새로운 위성 개발에도 쉽게 대응할 수 있는 기반을 마련하였다.

앞으로 RT-Linux 를 이용하여 시뮬레이터를 구축하거나, 위성 S/W 의 개발에 필요한 다양한 시뮬레이터를 보다 용이하게 개발할 수 있는 공통 Platform 구축도 가능하리라 생각한다.

참고문헌

[1] Power Control & Distribution Unit (PCDU) Equipment Specification, KARI
[2] CDI UART Code Walkthrough, 최종욱, KARI
[3] “Java AWT: Delegation Event Model”, http://java.sun.com/j2se/1.3/docs/guide/awt/designspec/events.html