인공 위성 운용 시스템의 실시간 처리 성능을 분석하기 위한 프로그램

하성준, 김소연, 한경숙
인하대학교 공과대학 자동화 공학과

A Program for Analyzing the Real-time Processing Performance of the Satellite Operation System

Sungjun Ha, Soyion Kim, Kyungsook Han
Automation Engineering Department, Inha University

요약

다목적위성 관제시스템의 한 부분인 위성운용 서브시스템의 실시간 처리 성능을 분석하기 위한 알고리즘과 이를 구현한 프로그램이 개발되었다. 이 프로그램은 위성운용 서브시스템에서 발생되는 event들과 이에 대한 response들의 속성에 관한 파라미터 값이 입력되었을 때, 각 event의 반응 시간과 스케줄 가능성을 계산하고 스케줄된 event들을 시각화한다. 실험결과, event의 blocking delay, 주기 및 첫 action의 우선 순위가 해당 event 또는 다른 event의 반응 시간에 많은 영향을 준다는 것이 밝혀졌다.

1. 서론

1999년 발사를 목표로 개발되고 있는 다목적 실용위성 (KOMPSAT: Korea Multi-Purpose Satellite)은 350~500kg, 고도 400~800km 정도의 저궤도 경량 위성으로, 한반도 관측 및 지도 제작, 과학 실험 등에 활용될 예정이다. 61일을 관측 주기로 갖는 다목적위성은 하루에 약 14.6 번 지구를 돈다. 관제시스템이 다목적위성과 교신하는 것은 위성이 한반도를 통과할 때만 가능한데, 매회 약 15분씩 하루에 4~6 번이다. 교신하는 동안 위성으로부터의 원격측정 데이터 (telemetry data)는 1초당 1개의 frame을 받는 속도로 수신된다.

위성운용 서브시스템 (SOS: Satellite Operations Subsystem)은 다목적위성의 관제시스템을 구성하는 네 개의 서브 시스템 (TTC 서브시스템, 위성운용 서브시스템, 임무분석 서브시스템, 위성 시뮬레이터 서브 시스템) 중의 하나이다. SOS는 위성이 한반도를 통과할 때 위성으로부터 수신한 원격측정 데이터를 실시간으로 처리하고 분석함으로써 위성의 상태를 감시하고, 위성으로 원격명령 (telecommand)을 송신함으로써 위성을 제어한다. SOS는 위성과의 교신 모드와 비교신 모드의 두 가지 모드로 동작하며, 이에 따라 SOS의 구성 소프트웨어는 on-line 소프트웨어와 off-line 소프트웨어로 구별된다. on-line 소프트웨어는, TTC (Telemetry, Tracking, and Command)로 수신된 원격측정 데이터를 실시간으로 처리하는 소프트웨어이고, off-line 소프트웨어는 on-line 소프트웨어가 사용하는 데이터 베이스를 구성하는 프로그램들이다. on-line 소프트웨어의 실시간 처리 성능은 관제시스템은 물론이고 위성 시스템 전체의 효율적 운용과 직결된다 [Garner and Jones 90]. SOS는 현재 3~4개의 같은 종류의 workstation이 LAN으로 연결되어 있고, task들이 특정 workstation에 static하게 bind된 분산 실시간 시스템으로 설계되고 있다.

본 논문은 저궤도 인공위성 관제시스템의 성능을 보다 체계적으로 분석하기 위한 노력의 일환으로서, 관제시스템의 위성운용 서브시스템의 실시간 처리 성능을 예측하기 위한 파라미터를 도출하고 이 파라미터들의 상관 관계를 분석하는 프로그램의 구현을 논한다.

2. 성능 분석 모델

2.1 관련 파라미터

SOS의 실시간 처리 성능을 분석하기 위한 파라미터는 두 가지로 분류할 수 있다: (1) 성능의 척도로서의 파라미터와 (2) 성능의 척도를 계산하는데 요구되는 파라미터. 성능의 척도가 되는 파라미터는 반응 시간 (response time), 스케줄 가능성 (schedulability)과 안정성 (stability)으로서, 이 중에서 반응 시간이 가장 중요한 파라미터라고 할 수 있다. 이러한 파라미터의 값을 결정하는데 요구되는 파라미터들에는 (1) event의 속성에 관한 파라미터 (event의 발생 원인, 도착 유형, 시간 제약 조건, blocking delay, deadline, mode), (2) response /action의 속성에 관한 파라미터 (response를 구성하는 action의 개수와 수행 순서, 사용하는 자원과 사용 시간, 우선 순위, jitter 요구 사항), (3) resource의 속성에 관한 파라미터 (resource의 유형과 할당 방식)가 있다.

2.2 성능 분석 알고리즘

SOS의 실시간 성능 분석을 위해 제안한 알고리즘은, 단일 프로세서에서 수행되는 실시간 시스템을 위하여 [Harbour *et al.* 91]과 [Klein *et al.* 93]에서 논의된 알고리즘을 다중 프로세서에서 수행되는 분산 실시간 시스템의 분석을 위하여 확장 개선한 것이다. 이 알고리즘은 다음과 같은 가정을 한다.

- 모든 event는 주기적이다.
- 모든 resource는 fixed-priority, preemptive policy에 의해 할당된다.

또한, 이 알고리즘은 다음과 같은 일반적인 경우에 적용 가능하다.

- event의 deadline에 대한 제한이 없다. 즉, deadline이 주기보다 작거나, 같거나, 또는 커도 된다.
- event에 임의의 blocking이 포함되어도 된다.
- 한 event에 대한 response의 우선 순위가 변경될 수도 있고, 사용하는 resource가 변경될 수도 있다. response의 우선 순위가 바뀌거나 사용하는 resource가 변경되면, 이들은 별개의 action으로 된다

이 알고리즘은 각 event가 사용하는 resource마다 Rate Monotonic Analysis (RMA)를 적용하여 event의 end-to-end response time (event의 모든 resource에서의 반응 시간의 합)을 계산하고 이를 그 event의 deadline과 비교하여 스케줄 가능성을 결정한다. 계산되는 반응 시간은 최악의 반응 시간 이다. 이 알고리즘에서 사용되는 용어에 대해서 설명하면 다음과 같다.

- P_{ij}: event ei의 j번째 action의 우선 순위
- Level-P_i busy period: 프로세서가 P_i보다 우선 순위가 높거나 같은 job들을 처리하느라 바쁜 기간 (Lehoczky 90)
- Canonical form: response의 action들의 우선 순위가 감소하지 않는 순서대로 있는 형태
- H(j): P_{ij}보다 크거나 같은 우선 순위를 갖는 action들로만 이루어진 event들의 집합 (ei는 제외)
- L(j): P_{ij}보다 작은 우선 순위를 갖는 action들로만 이루어진 event들의 집합 (ei는 제외)
- HL(j): P_{ij}보다 높거나 같은 우선 순위를 갖는 action으로 시작하여 P_{ij}보다 작은 우선 순위를 갖는 action이 나중에 존재하는 event들의 집합 (ei는 제외)
- LH(j): P_{ij}보다 낮은 우선 순위를 갖는 action이 존재하고, P_{ij}보다 크거나 같은 우선 순위를 갖는 action이 연속적으로 존재하는 event의 집합 (ei는 제외)
- m(i): event ei를 canonical form으로 고친 후, ei의 action의 개수
- $R_r(e_i)$: resource r에서의 대한 ei의 반응 시간

그림 1은 알고리즘을 간략히 요약하고 있다.

1. $R_r(e_i)$와 R(ei)를 0으로 초기화한다.
2. ei를 canonical form으로 고친다.
3. H(1), HL(1), L(1), LH(1)에 해당되는 event 결정한다.
4. Level-P_{i1} busy period를 구한다.
5. N: Level-P_{i1} busy period안에 jobs의 개수
6. 0으로 초기화된 job counter b를 하나 증가. b와 N과 비교, b>N이면 12번 단계로 간다. 아니면 7번 단계로 간다.
7. ei의 b번째 job의 첫 번째 action의 completion time을 계산한다.
8. 1로 초기화된 j를 하나 증가하여 m(i)와 비교, j>m(i)이면 completion time을 E(b)에 대입, 6번 단계로 가고, j≤m(i)이면 9번 단계로 간다.
9. H(j-1)를 H(j)와 HL(j)로 다시 구분한다 (j >1)
10. ei의 b번째 job의 j번째 action의 completion time을 계산
11. 8번 단계로 간다.
12. $R_r(ei) = \underset{1\le b\le N}{MAX}\{E(b)-(b-1)T_i\}$를 구한다.
13. $R(e_i) = R(ei) + R_r(ei)$를 구한다.
14. R(ei)와 ei의 deadline 비교하여 schedule의 여부를 결정한다

그림 1. 성능 분석 알고리즘

3. 프로그램 구현 및 실험결과

이 알고리즘은 PC/Window 95 환경에서 C++ Builder로 구현되었다. 앞에서 언급된 제 2부류의 파라미터 값이 입력되면, 각 event의 최악의 반응 시간과 스케줄 가능성이 계산된다. 또한 스케줄된 event들에 대한 time line sketch도 보여준다 (그림 2).

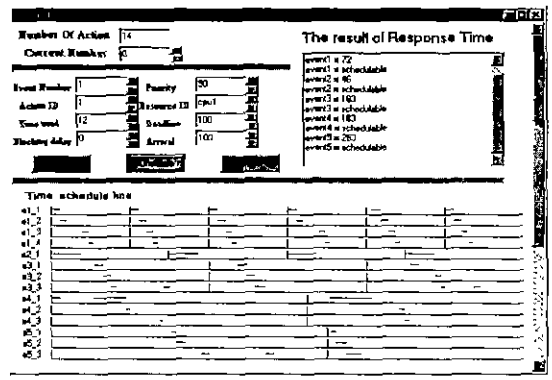

그림 2. 성능 분석 프로그램

예를 들면, 표 1의 실시간 상황이 입력되었을 때 event e1, e2, e3, e4, e5의 반응 시간은 각각 72, 46, 183, 183, 283 ms로 계산되고, 이들 모두 스케줄 가능하다고 출력된다.

action ID	blocking delay	period	deadline	event name	resource ID	time_ used	priority
a1_1	0	100	100	e1	cpu1	12	90
a1_2	0	100	100	e1	cpu1	10	100
a1_3	0	100	100	e1	cpu1	8	150
a1_4	0	100	100	e1	cpu1	6	110
a2_1	20	150	150	e2	cpu1	18	130
a3_1	0	200	150	e3	cpu1	12	30
a3_2	0	200	200	e3	cpu1	10	60
a3_3	0	200	200	e3	cpu1	14	40
a4_1	40	325	325	e4	cpu1	13	100
a4_2	0	325	325	e4	cpu1	7	50
a4_3	0	325	325	e4	cpu1	5	120
a5_1	0	350	350	e5	cpu1	7	10
a5_2	0	350	350	e5	cpu1	14	40
a5_3	0	350	350	e5	cpu1	21	20

표 1. 실시간 상황의 예

3.1 우선순위와 반응 시간의 관계

표 1의 실시간 상황에서 e1의 첫 번째 action의 우선순위를 변경했을 때, 그림 3과 같은 결과가 나왔다.

그림 3. P_{11}의 변경에 따른 event들의 반응 시간

위의 결과를 보면 P_{11}이 높을수록 e1의 반응 시간은 감소하거나 변화가 없지만, 다른 event의 반응 시간은 늘어난다. 특히 우선순위가 50, 120에서는 e4, 30에서는 e3, 130에서는 e2기 급격히 변함을 알 수 있다. 이 우선순위 값들은 각 event가 canonical form으로 고쳐졌을 때 그 event의 action들의 우선순위 값이다. 이것은 P_{11}이 커짐에 따라 e1에 대한 H(1), HL(1)가 L(1) 또는 LH(1)로 변화하는 경우가 생기기 때문이다. P_{11}의 변화는 P_{12}, P_{13}의 변화보다 ei 및 다른 event들의 반응 시간에 영향을 준다는 것을 알았다.

3.2 주기와 반응 시간과의 관계

표 1의 실시간 상황에서 e3의 주기를 변경하면, 각 event들의 반응 시간은 그림 4와 같이 변한다. 이 결과를 보면 e3는 e5와 비교할 때 H(1)에 해당된다. e3의 주기가 증가하면 e5의 반응 시간이 줄어드는 결과를 볼 수 있다. 하지만 e3는 e1, e2에 대해 L(1)에 해당되고, e4에 대해서는 LH(1)에 해당되기 때문에, 결과는 아무 영향을 주지 못했다. 즉, e3를 H(1)로 여기는 event들 (e5)만이 영향을 받는다. e3의 주기의 변화는 event e1, e2, e4에 아무런 영향을 주지 못한다.

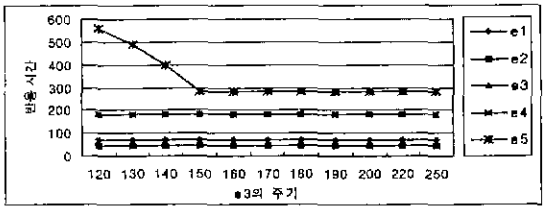

그림 4. e3의 주기변화에 따른 event들의 반응 시간

3.3 Blocking delay와 반응 시간과의 관계

표 1의 실시간 상황에서 e4의 blocking delay를 변경하면서 다른 event들의 반응 시간은 그림 5와 같다. e4의 blocking delay는 다른 event의 반응 시간에 영향을 주지는 않지만 자기 자신의 반응 시간에는 민감한 영향을 준다. blocking delay가 증가하면 그 반응 시간이 전체적으로 linear하게 증가하지만 어느 순간에서는 급격히 증가하는 것을 볼 수 있다. e4가 canonical form고쳤을 때 H(1)에 해당되는 event들이 없다면 linear하게 증가할 것이다.

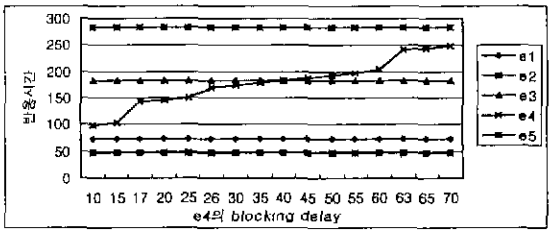

그림 5. e4의 blocking delay의 변화에 따른 event의 반응 시간

3.4 action의 개수와 반응 시간과의 관계

event의 action의 개수가 반응 시간에 미치는 영향을 분석하기 위해, 표 1에서 3개의 action으로 구성된 event e3의 수행 시간을 본래 3개의 action의 수행 시간 합인 36 (12+10+14)로 하고, 하나의 action으로 변경하였다. 변경된 상황에 대한 결과는 그림 6에서 보듯이, e1, e2, e5는 아무런 영향을 받지 않았다. 하지만 e4는 e3의 우선 순위의 변화에 영향을 받는 것을 알 수 있다. 원인을 분석하면 e3는 다른 event를 기준으로 H(1), HL(1), LH(1), L(1)으로 4종류로 구분된다. e3의 우선 순위가 30으로 바뀌면 e4에 대해 e3는 LH(1)에서 L(1)으로 되기 때문에 e4가 반응 시간이 줄어드는 것이다. 즉 e3의 우선 순위가 바뀌면 HL(1) 또는 LH(1)에 해당되는 event는 H(1) 또는 L(1)으로 바뀌므로 반응 시간에 변화가 생기는 것이다. e3를 H(1) 또는 L(1)으로 구별시키는 event들은 e3의 우선 순위가 30, 60 또는 40인 하나의 action으로 변경하여도 그 event들의 반응 시간은 아무런 변화가 생기지 않는다.

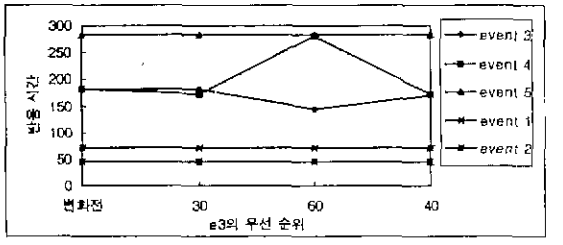

그림 6. e3의 우선 순위에 따른 다른 event들의 반응 시간

4. 결론 및 향후 연구

본 논문은, 다목적 실용위성 위성운용 시스템의 실시간 처리 성능을 분석하기 위한 알고리즘과 이를 구현한 프로그램을 소개했다. 구현된 프로그램을 이용하여 실험적인 방법으로 특정 파라미터들 (blocking delay, 주기, 수행시간, 우선 순위)과 반응 시간과의 관계를 규명했다. 실험 결과를 요약하면, event ei의 blocking delay의 변화는 ei의 반응 시간에만 민감하게 영향을 주고 다른 event들에게는 아무런 영향을 주지 않았다. ei의 주기의 변화는 ei의 반응 시간에 그다지 영향을 주지는 못함을 알 수 있었고, ei의 첫 번째 action의 우선순위의 변경은 후속하는 action의 우선 순위의 변경보다 ei와 다른 event의 반응 시간에 더 많은 영향을 주는 것을 알았다. 파라미터 값의 상관 관계 분석 결과를 사용자에게 조언하는 부분을 프로그램에 첨가하기 위한 노력이 진행중이다. 즉, 프로그램 수행 결과, 반응 시간이 deadline보다 커서 스케줄 가능하지 않다든지 spare capacity (즉, 그 event보다 우선 순위가 낮은 event들의 스케줄 가능성을 유지하면서 그 event에 추가할 수 있는 수행 시간)가 너무 크면, 어느 파라미터 값을 어느 정도 증감해야 하는지를 사용자에게 조언하는 부분을 프로그램에 포함시켜서 더욱 유용한 성능 분석 프로그램이 되도록 할 것이다.

참고문헌

J. T. Garner and M. Jones. *Satellite Operations: systems approach to design and control*, Ellis Horwood, 1990.

M. G. Harbour, M. H. Klein, and J. P. Lehoczky. Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority. *Proceeding of the IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, 1991, 116-128.

M. H. Klein, T. Ralya, B. Pollak, R. Obenza, and M. G. Harbour, *A Practitioner's handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.

J. P. Lehoczky. Fixed Priority Scheduling of Period Task Sets with Arbitrary Deadlines. *Proceedings of the IEEE Real-Time Systems Symposium*, IEEE Computer Society Press, 1990, 201-209.