실시간 SOA에서 서비스의 실행시간 예측

(Service Execution Time Estimation in Real-time SOA)

김 여 자 † 변 정 용 ††
(Yeoja Kim) (Jeongyong Byun)

요 약 기존의 실시간 시스템들이 SOA기반으로 통합되면 실시간 SOA가 필요하다. 실시간 SOA에서 하나의 서비스는 일반적으로 여러 개의 작은 단위서비스로 분화되기도 한다. 단위서비스의 예측실행시간은 제공자시스템에서 제공된다. 하지만 요청자는 중개자와 제공자 사이의 메시지 송수신 과정에 관여되는 시간요소를 분석한 시간예측도 필요하다. 본 논문은 웹 서비스의 QoS 만족도를 높이기 위해서 트랜잭션을 고려한 시간 예측과 더불어 다중처리기 시스템의 보편화에 따른 트랜잭션의 실행시간 예측에 관련된 시간요소를 분석하여 개선된 최악의 실행시간 예측 방법을 제안한다.

키워드 : 최악실행시간과 처리기수의 관계, CPU실행시간 분석, DB 실행시간 분석

Abstract If the existing real-time systems are integrated based on SOA, real-time SOA should be developed. Generally, in real-time SOA a service can be divided into several small services and their estimated execution time is given by provider systems. However, an estimation, which analyzes time elements related to transmit and receive messages among requesters and providers, is needed. In order to enhance QoS of Web service, this paper proposes enhanced worst-case execution time estimation by considering WS-transaction and common use of multi-processors system.

· 본 연구는 교육과학기술부와 한국산업기술재단의 지역 혁신 인력양성사업으로 수행하였습니다.
· 이 논문은 2008 학술심포지움에서 '실시간 SOA에서 서비스의 실행시간 예측'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 동국대학교 전자계산학과
sola@dongguk.ac.kr
†† 종신회원 : 동국대학교 컴퓨터멀티미디어학과 교수
byunjy@dongguk.ac.kr

논문접수 : 2009년 3월 3일
심사완료 : 2009년 4월 28일

Key words : Relation of Worst-case execution time and processors, analysis of CPU execution time, analysis of DB execution time

1. 서 론

오늘날 기업들은 내부시스템들의 통합, 기업간거래에 따른 기업업무통합운용, 다국적 기업업무통합 등 다양한 변화 및 투자 효율 극대화를 위해 코드 재사용과 이 기종 기업간의 상호운용[1]을 가능하게 해주는 SOA(Service Oriented Architecture)을 적용하고 있다. 여기서 통합 대상 시스템들이 실시간 시스템일 경우 실시간 처리가 지원되어야 한다.

이러한 SOA시스템에서 실시간 처리를 지원하기 위해서는 기존 실시간 시스템에서 분석된 시간요소들[2-4]과 네트워크에서 소요되는 시간요소, 그리고 SOA환경에 의해서 추가된 WSDL(Web Service Description Language)를 참조하여 SOAP(Simple Object Access Protocol) 메시지를 작성[5, 9-13]하는 시간요소들이 필요하다. 실시간 SOA에서 웹 서비스 트랜잭션(WS-transaction)을 서비스하는 시스템의 경우도 실시간 트랜잭션을 서비스하기 위해서는 실시간을 지원하는 시간요소가 필요하다. 필요한 시간요소들이 네트워크로 연결된 제공자 시스템의 CPU, DB(Database)등에서 실행하는 시간요소들은 대기 큐에 있는 SOAP메시지들의 개수를 알 수 없기 때문에 제공자 시스템의 시간요소들은 분석하여 실시간으로 제공해주는 서비스가 필요하다. 또한 실시간으로 분석된 시간요소를 제공하기 위해서는 제한된 환경에서 낮은 대기시간과 높은 처리율, 그리고 긴급함이 다른 제한사항을 다르게 처리[6]할 수 있는 알고리즘이 필요하다. 그리고 사용 알고리즘과 다중처리기에 따른 응답시간도 비교분석이 필요하다.

본 연구에서는 서비스의 QoS향상을 위해 예측실행시간으로 중개시스템의 시간요소들, 네트워크 시간요소, 제공자 시스템에서 시간요소들을 종합하여 제공하고, 종합된 시간요소들 중에서 제공자시스템의 CPU와 DB 실행시간에 대한 시간요소들은 실시간 최악실행 시간으로 분석하여 제공한다. 그리고 사용 알고리즘과 다중처리기에 따른 응답시간 등도 비교 분석하여 제공하고자 한다.

2. 관련 연구

실시간 트랜잭션 메시지를 SOA환경으로 확장하여 수행하기 위해서는 실시간 시스템에서 사용한 최악 시간 분석 응답시간[2]과 태스크 모델[4], 네트워크 소요된 시간 계산식[7,8] 그리고 SOA에서 실시간 트랜잭션을 실행하기 위한 환경[4,5,9-14], 웹 서비스 성능분석을 위한 시간요소 등이 참조된다[15].

2.1 실시간 시스템에서 최악응답시간 분석

[1]에 의한 최악응답시간 분석에 의하면 실시간 시스템의 단일 처리기에서 최악실행시간은 식 (1)과 같다.

$$R_i = C_i + B_i + I_i \quad (1)$$

R_i : 최악응답시간, C_i : CPU실행시간

B_i : 블록킹 시간, I_i : 방해 시간을 의미한다.

2.2 실시간 태스크 모델

[3]에 의하면 온라인 실시간 태스크 모델은 LLA (Least Laxity Algorithm), EDZL/n-2 등이 있다. EDZL/n-2는 다중처리기 알고리즘으로 그림 1에서 간략하게 표현하고 있다.

```
EDZL/n-2 알고리즘
  While (인터넷에서 메시지 도착){
    if (여유시간 <= n-2)
      LLA정책 실행;
    Else
      EDA정책 실행;}
```

그림 1 EDZL/n-2알고리즘

2.3 네트워크 시간요소

기존에 연구된 네트워크에서 소요된 시간을 계산하기 위한 계산식을 보면 A라우터에서 B라우터까지 가는데 걸린 시간은 $D_{nodal}=(d_{proc}+d_{queue}+d_{trans}+d_{prop})*m$, m: 패킷 수이다. 라우터 수가 n이라면 네트워크에서 소요된 시간은 = n* D_{nodal} [7,8]이다.

2.4 웹 서비스 트랜잭션(WS-Transaction)

트랜잭션이 고정적인 시스템에서 사용되다가 실시간 시스템에서 사용되어지기 위해 사용가능한 환경으로 확장 되어야한다[2]. 실시간 트랜잭션들이 SOA 환경에서 적용되기 위해서는 웹서비스의 확장 스팩중에서 웹서비스 트랜잭션(WS-Transaction) 스팩을 사용한다[4,5, 9-14]. 웹 서비스의 성능분석을 위한 시간요소는 네트워크 트랜잭션시간, 메시지 처리시간, 서비스 실행 시간으로 나누어진다[15].

3. 트랜잭션 SOAP 메시지 처리시스템 설계

실시간 SOA에서 QoS 만족도를 높이기 위한 실시간 트랜잭션 SOAP 메시지를 처리하는 중개시스템과 제공자시스템에 대해서 설계하고. 우리는 [15]를 7개 시간요소(㉠ ㉡ ㉢ ㉣ ㉤ 식 (2) 식 (3))로 분리하여 제공하고자 한다. 그리고 이후에 나타나는 실시간 트랜잭션 SOAP 메시지를 '메시지'라 지칭한다.

그림 2의 중개시스템은 웹 서비스 중개시스템을 표현한 것으로 단일 프로세서 시스템환경이라 가정한다. 여러 요청자가 중개시스템에 접근하여 서비스를 요청하면 서비스를 작은 단위서비스로 분화하여 각각을 제공자

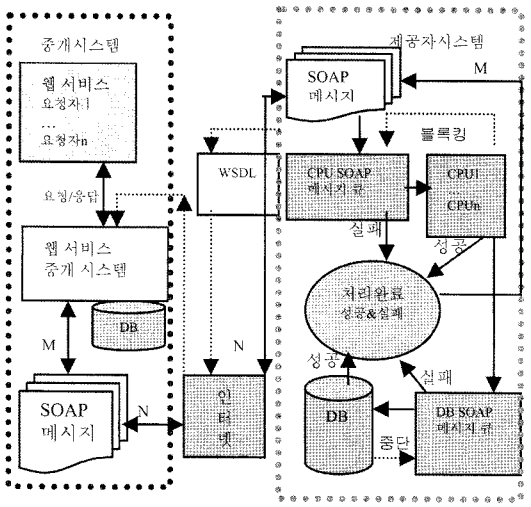

그림 2 중개시스템과 제공자시스템

시스템에 메시지로 보낸 다음 응답메시지가 오면 응답을 분류 종합하여 요청자에게 제공해 주는 시스템이다. 그리고 서비스에 대한 예측실행시간을 요청하면 단위서비스를 제공자시스템으로 보낸 후에 표 4의 시간과 제공자 시스템에서 보내온 표 5의 응답시간을 참조하여 요청자에게 예측실행시간을 제공해준다.

그림 2의 제공자시스템은 웹 서비스를 제공하는 제공자시스템을 표현한 것으로 다중처리기로 구성되어 있고 하나의 메시지는 한 단위시간에 여러 개의 CPU에서 실행되지 않고 오직 하나의 CPU에서만 처리되고 DB는 분산 DB가 아닌 제공자시스템에 있는 DB를 사용하여 처리한다고 가정한다. 중개시스템으로부터 서비스가 도착되면 예측 실행시간이 필요한 서비스는 도착된 메시지와 기존 대기 큐에 있는 메시지를 프로그램적으로 수행한 후 제공자시스템의 예측실행시간으로 중개시스템에 전달한다. 그리고 서비스를 처리한 후 각 중개 시스템에 처리 완료된 응답메시지를 보내는 시스템이다.

4. 서비스에 대한 시간요소 분석

[15]를 더 작게 나누면 중개시스템에서 시간 요소는 ㉠: 네트워크 소요시간, ㉡: 메시지 작성시간, ㉢: 응답메시지 분류종합시간 등이 있고 제공자시스템에서 시간요소는 ㉣: 응답 메시지 작성시간, ㉤: 응답 메시지 네트워크 소요시간, 식 (2): CPU 실행시간, 식 (3): DB실행시간 등이 있다. 이 요소들 중에서 CPU, DB실행시간은 최악실행시간으로 분석하여 제공하고 나머지 시간요소는 실제시간으로 분석한다. 그리고 분석된 시간요소들을 종합하여 분석된 예측실행시간을 제공하고자 한다.

4.1 CPU와 DB 최악실행시간 분석

CPU 수행시간은 다중처리기 알고리즘인 EDZL/n-2을 사용하여 분석한다. 이 알고리즘은 매 단위시간마다 우선순위를 체크하여 문맥교환을 하기 때문에 B_i시간은 없는 것으로 간주한다. 우리는 기존의 단일 처리기일 때 적용되는 식 (1)을 다중처리기 시스템으로 변환하여 분석한다. 표 1의 자료를 가지고 표 2의 응답시간들을 비교해보면. 자신의 수행시간은 식 (1)같지만 I_i시간은 처리기수 증가에 따라 I_i/n으로 감소함을 알 수 있다. 따라서 다중처리기 사용에 따른 CPU 최악응답시간은 식 (2)가 된다.

$$RM\_Cpt_i = C_i + I_i,$$
$$\text{단 } \frac{\lfloor I_i \rfloor}{n} - 1 <= I_i < \frac{\lfloor I_i \rfloor}{n} + 1 \quad (2)$$

DB 실행시간을 분석하기 위해서 식 (3)이 사용된다. 식 (3)은 DB 메시지 큐에 있는 메시지가 실행되는데 걸리는 시간을 수식으로 표현한 것으로 식 (1)의 블록킹 시간을 이 논문에서 조건이 있는 블록킹 시간으로 확장한 것이다. 따라서 DB 최악응답시간은 식 (3)이 된다.

$$RM\_DBpt_i = DBp_i + B_i + I_i + I_{i\_abortime} \quad (3)$$

DB_{pi} : 트랜잭션 SOAP 메시지가 실행된 시간

$$B_i = \sum_{k=1}^{k} usage(k,j)_{j \in lp(i) \wedge xlock \wedge (hrlt \geq etrt(j))} DBp(k) \quad (4)$$

$$I_i = \sum_{j \in hp(i)} \left\lceil \frac{R_i + J}{T_j} \right\rceil DBp_j \quad (5)$$

$$I_{i\_abortime} = \sum_{j \in hp(i) \wedge (hrlt(j) < etrt(i))} DBp_{i\_execution\_time} \quad (6)$$

hrlt : 높은 우선순위 트랜잭션의 여유시간

etrt : 실행 중인 낮은 우선순위 트랜잭션의 남은 실행시간

식 (4)는 자신보다 낮은 우선순위를 가진 메시지가 처리될 때까지 기다린 시간으로 이 시간은 우선순위가 높은 메시지가 여유가 있을 때만 가능하다. 식 (5)는 기존 식이고 식 (6)은 식 (4)의 반대 조건으로 DB실행하다가 중단되기까지 실행된 시간들의 합을 나타낸 것이다. 식 (4)와 식 (6)은 논문에서 트랜잭션 실행 중단을 최대한 방지하고 자원낭비를 최소화하기 위해 제안된 식이다.

4.2 그 외의 시간요소 분석

그림 2에서 N은 네트워크에서 소요된 시간을 의미하며 작은 단위 서비스이므로 하나의 패킷으로 만들어져 네트워크를 통과한다고 가정한다. 우리는 라우터마다 도착하는 메시지 수가 다를 수 있기 때문에 각각의 라우터에서 소요된 시간이 다를 수 있다. 따라서 중개시스템에서 기존 식을 변형하여 네트워크에서 소요된 시간은 식 ㉠이라 한다.

$$M\_nt_i = \sum_{i=1}^{n} d_{nodal} = \text{네트워크(출발시간} - \text{도착시간)} \quad ㉠$$

네트워크 소요시간만 ㉠과 같이 계산하여 사용하고 그 외의 요소들은 모두 기존 라우터에서 사용된 방법들을 그대로 사용한다.

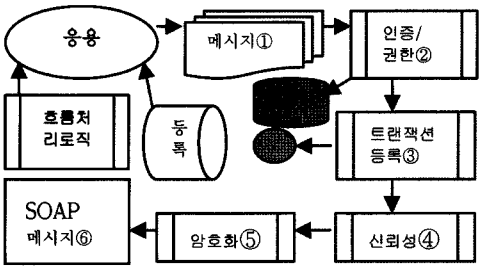

그림 3 트랜잭션 SOAP 메시지 작성과정

그림 3은 그림 2의 M을 표현한 것으로 메시지 작성하는 과정을 그림으로 표현한 것이다[5]. 그림 3에 따라서 중개시스템에서 메시지 작성 하는 시간은 식 ㉡이라 한다.

M_wt_i=메시지 네트워크 출발시간-서비스 요청시간 ㉡

그리고 제공자시스템으로부터 응답 메시지가 도착되면 분류 종합하여 요청자에게 제공해주는 중개시스템의 시간요소는 식 ㉢이라 한다.

RM_Rpt_i=응답서비스 제공시간-응답 메시지 도착시간 ㉢

제공자시스템에서 처리가 모두 완료된 서비스에 대해 응답메시지 작성시간은 ㉣이고, 응답메시지가 네트워크에서 소요된 시간은 ㉤이다

RM_wt_i=네트워크 출발 시간
- SOAP 메시지 실행 완료 시간 ㉣

RM_nt_i=중개시스템 도착시간
- 제공자시스템 출발 시간 ㉤

4.3 서비스의 예측실행시간

중개시스템과 제공자시스템의 시간요소들을 모두 종합하면 식 (7)이 된다.

$$R_i = ㉠+㉡+㉢+㉣+㉤+\text{식(2)}+\text{식(3)} \quad (7)$$

식 (7)은 하나의 단위서비스가 응답을 받을 때까지 시간을 수식으로 표현한 예측실행시간이다.

5. 서비스의 예측 실행시간 실험

먼저 4장에서 분석된 CPU의 처리기 수에 따른 최악응답시간과 DB 최악응답시간을 실험한 후 나머지 시간요소들은 기존 논문[16]에서 사용한 실험에 시간요소를 추가하여 실험한 실제시간으로 서비스에 대한 예측실행시간을 실험한다. 또한 CPU분석을 위해서 사용된 알고

리즘은 온라인에서 최적인 알고리즘과 처리율, 문맥교환, 실행시간 등을 비교 분석한 결과를 제공하고자 한다.

5.1 CPU와 DB 최악응답시간 실험

실험을 위한 환경은 Windows XP Professional Version2002, CPU은 Intel® Pentium® 4 CPU 2.80GHz, 메모리는 1.50GB RAM, Hard는 Intel(R) 82801 Ultra ATA Storage이고 비주얼 스튜디오 2005에서 C++ 언어를 사용하여 구현했다. 다중처리기 사용은 오래전부터 다중 처리기사용에 대한 연구가 계속되어 왔고 최근에는 서버용으로 사용되는 대부분의 컴퓨터들이 다중처리기를 채용하고 있기 때문이다[16].

표 1은 CPU큐에 있는 메시지를 나타낸 것으로 DB 실행이 필요한 메시지는 DB수행에 표시가 되어있다고 가정한다. 표 1을 처리기 수에 따른 응답시간으로 실험하여 표 2에 나타냈다.

표 1 CPU 큐에 있는 SOAP 메시지

도착 시간	CPU time	CPU 종료시한	여유 시간	DB수행
0	4	4	0	T1
0	4	4	0	T2
0	4	4	0	T3
0	4	4	0	.
0	7	10	3	.
0	7	10	3	.
0	7	10	3	.
0	7	10	3	.
0	1	7	6	.

표 2를 보면 처리기 수가 1~4개까지는 종료시한을 초과해서 처리되는 메시지수가 있지만 처리기 수가 5개 일 때부터는 모두 다 종료시한을 만족하여 처리된다. 따라서 처리기수가 5개일 때 최악수행시간을 나타낸 것이고 최적 수행시간은 문맥교환이 없는 처리기 수가 9개일 때이다.

이렇게 CPU에서 실행을 마친 메시지들은 다시 DB에 있는 경쟁 있는 데이터를 접근하기 위해 DB SOAP 메시지 큐에서 대기한다. 대기 큐에는 표 3이 주어진다.

표 2 처리기수에 따른 응답시간

처리기수 / 메시지	1	2	3	4	5	6	7	8	9
(4,4)	13	7	5	4	4	4	4	4	4
(4,4)	14	7	5	4	4	4	4	4	4
(4,4)	15	8	5	4	4	4	4	4	4
(4,4)	16	8	6	4	4	4	4	4	4
(7,10)	42	21	14	11	10	9	8	7	7
(7,10)	43	22	15	11	10	9	8	7	7
(7,10)	44	22	15	11	10	9	8	7	7
(7,10)	45	23	15	12	10	9	8	8	7
(1,7)	29	15	10	8	5	5	5	4	1

표 3 DB 큐에 있는 SOAP 메시지들

	T(주기)	DB 시간	Deadline	도착시간
T1	15	8	10	0
T2	25	5	25	0
T3	30	5	30	0

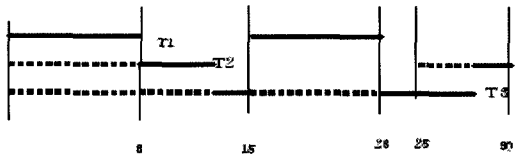

그림 4 DB 큐에 있는 메시지 처리 과정

표 3의 DB수행과정을 그림 4에 나타냈다.

그림 4에서 응답시간은 T_1: 8, T_2: 13, T_3: 28이고 점선은 식 (5)를, T_3의 14-15의 2단위는 식 (6)을 나타낸다.

5.2 그 외의 시간요소 실험 및 예측실행시간

그 외의 시간요소를 나타내는 표 4는 웹 서비스 요청에 의해 제공자시스템의 DB에 있는 데이터를 가져와서 결과를 보여주는 실험[17]에 시간요소만 추가하여 10번 수행 후 식을 간략화하기 위해 가장 오래 걸린 수행시간을 10으로 나누어 반올림한 최악응답시간이다(단위: ms).

표 4 실험 분석된 최악실행시간

㉠	㉡	㉢	㉣	㉤
2	8	0	0	2

㉢의 0은 받은 메시지를 변수에 담아서 화면에 뿌려주는 시간이고 ㉤의 0은 DB실행 후 변수에 담는 시간이다.

표 5는 CPU와 DB에서 분석된 최악응답시간이다.

표 5 CPU와 DB 최악실행시간

	T_1	T_2	T_3
CPU	4	4	4
DB	8	13	28

시간단위는 다르지만 같은 시간단위라 가정한다. 따라서 단위서비스들의 최악예측실행시간은

R_{T1} : 2+8+0+0+2+4+8=24

R_{T2} : 2+8+0+0+2+4+13=29

R_{T3} : 2+8+0+0+2+4+28=44가 된다.

5.3 알고리즘 비교 분석

온라인에서 최적인 LLA와 EDZL/n-2을 비교 분석하고자 한다. 객관성을 주기 위해 rand()함수를 사용하여 CPU 시간과 종료시한 값을 할당했으며, 비교하는 대상은 처리기수에 따른 성공횟수, 성공문맥교환횟수 그리고 알고리즘 수행시간이다. 표 6은 알고리즘을 비교하기 위한 실험조건을 나타낸 것이다.

표 6 알고리즘 비교 실험조건

메시지 수	1000	주기당 메시지 수	3개
처리기수	2,5,10,15	Cpu시간	1~5
주기	5	종료시한	0~15

표 7 처리기 수에 따른 성공횟수, 문맥교환횟수 비교

	EDZL/n-2		LLA	
처리기 수	성공 수	문맥교환 수	성공 수	문맥교환 수
2	537	49	519	89
3	795	160	792	229
5	868	203	868	211
7	911	188	911	199
10	935	168	935	168
15	960	134	960	134

표 6의 조건을 기반으로 처리기수에 따라 10번씩 수행한 평균값으로 성공횟수와 문맥교환횟수를 나타낸 것이 표 7이다.

그리고 표 8은 처리기 수가 5개일 때 10번 수행한 평균 실행시간이다. 단위는 밀리 초이다.

표 8 알고리즘 실행시간 비교

알고리즘	LLA	EDZL/n-2
실행시간	2.53298129049529/ms	2.80966366508483/ms

표 8에서 EDZL/n-2가 LLA 보다 시간이 0.27ms 더 걸린 것은 EDA수행처리와 LLA수행처리를 구분하기 위한 비교 구문이 있기 때문이다.

6. 결 론

우리는 실시간 SOA에서 서비스의 종료시한을 만족하는 예측실행시간으로 수식(7)을 제공했다. 제공된 시간요소 중에서 제공자시스템의 CPU와 DB 실행시간은 최악실행시간으로 분석하여 종료시한을 만족하도록 제공했다. 그리고 사용한 EDZL/n-2 알고리즘은 온라인에서 최적인 LLA알고리즘과 비교 분석하여 처리율은 같지만 문맥교환횟수는 EDZL/n-2알고리즘이 더 적다는 것을 표 7에서 성공횟수와 문맥교환횟수 실험을 통해서 보였고, 실행시간은 EDZL/n-2알고리즘이 표 8에 따르면 0.27ms 더 많이 걸린다. 문맥교환과 알고리즘 수행시간 중에서 우리는 문맥교환 오버헤드를 줄이는 방법으로 EDZL/n-2알고리즘을 사용했다. 또한 다중 처리기 사용은 표 1 데이터를 기반으로 표 2의 처리기수에 따른 응답시간 비교를 통해서 CPU에서 실행된 시간요소 중에서 방해시간이 CPU개수에 따라 적어짐을 알 수 있고 CPU 큐에 있는 메시지에 따라 최악실행시간을 실행하는 처리기 개수를 알 수 있다. CPU 실행 응답시간이 표 2의 문맥교환이 없는 최적응답시간에서 최악응답시간으로 실행되어진다면 관리자는 처리기 수 증가를 결정해야 함을 실험을 통해서 보였다. 이렇게 제공한 예측실행시간은 요청자는 대기시간에 참조하고 중개시스템은 흐름 조정이 가능한 스케줄, 종료시한, 타임아웃 속성에 사용할 수 있다. 또한 오래 수행되어지거나 데드락을 해결하기 위해서도 사용될 수 있다.

참 고 문 헌

[1] J.S.Choi, H.S.Kim, H.T.Kim, S.U.Mun, J.H.Lee, H.J.Lee, translation., *Service-Oriented Architecture A Planning and Implementation Guide for Business and Technology*, Mplanning, 2007.

[2] Alan Burns & Andy Wellings., *Real-Time Systems and Programming Languages*, Addison-Wesley, 2003.

[3] Byun JY, Andy Wellings and Alan Burns, "A Worst-Case Behaviour Analysis for Hard Transactions," *International Workshop of 96 RTDB*, U.S.A., vol.1, no.1, pp.150-155, 1996.

[4] Yeoja Kim and Jeongyong Byun, "A Real-time Task Scheduling for Multi-Processor System," *Proceedings of The 3rd on EALPIIT-2003*, Ultanbataar of Mongolia, pp.228-233, 2003.

[5] Eric Newcomer, Greg Lomow., *Understanding Soa with Web Services*, Addison-Wesely, U.S.A, 2005.

[6] SOA Feature Story : Real-Time SOA Start with the Messaging Bus, "http://events.sys-con.com/node/467488"

[7] James F. Kurose, Keith W.Ross., *computer networking*, Addison-Wesley, U.S.A, 2005.

[8] MTU?, http://blog.naver.com/ldbina/120051610083.

[9] Thomas Erl., *SOA Principles of Service Design*, Prentice Hall, U.S.A, 2008.

[10] Thomas Erl, *Service-Oriented Architecture A Field Gudie to Integrating XML and Web Services*, Prentice Hall, U.S.A, 2004.

[11] http://www.soaspecs.com

[12] S.Y,Jang, S.C.Whang, H.J.Lee, M.O.Cho Translation., *Service-Oriented Architecture :Concepts, Technology, and Desing*, Acorn, 2006.

[13] J.S.Park other 11 people Translation., Developing Enterprise Web Services An Architure's Guide, Hongrung, 2008.

[14] Samsung SDS SOA Research Translation., *Service-Oriented Architecture A Field Guide to Integranting XML and Web Services*, Sungandang, 2007.

[15] H.J.Na, D.J.Choi, "Design and Implementation of Performance Measurement System for Web Services," *KNOM Review*, vol.6, no.2, pp.13-20, 2004.

[16] http://devnote.net/wiki/index.php/The opening of MultiProcessor Programming period/. 2007.

[17] Aziz Nasridinov, KyoungWook Kim, JeongYong Byun, "A services Quality Enhancement of Business Process Using Web Services," *Proceedings of the 8th Apis*, Jan.11-12, pp.349-353, 2009.