플랫폼 독립형 컴포넌트를 기반으로 구축된 통신망 관리 시스템의 성능분석

박 수 현*○, 백 두 권**
* LG정보통신(주) 중앙연구소 응용교환실
** 고려대학교 컴퓨터학과 소프트웨어시스템 연구실

Performance Evaluation of Network Management System based on Platform Independent Component

Soo-Hyun Park*○, Doo-Kwon Baik**
* Advanced Switching Lab., Central R&D Center, LGIC. Ltd.
** Software System Lab, Dept. of Computer Science & Eng., Korea University

요 약

플랫폼 독립형 클래스 저장소(PICR : Platform Independent Class Repository)에 저장되어 있는 컴포넌트를 망관리 시스템의 분산객체로 아웃소싱하여 사용하는 Farming 방법론을 실제 TMN 에이전트의 구축에 적용하였을 때 나타나는 결과를 ATM과 같은 고속망을 DCN(Data Communication Network)으로 하는 경우에 대하여 성능평가를 하였다.

I. 서론

TMN (Telecommunication Management Network) [1][2][3][4] 은 구축 과정에서 여러 가지 서로 다른 운영체제 및 이 기종의 하드웨어 플랫폼을 이용하여 개발되어지는 관계로 TMN 시스템 에이전트의 클래스를 개발하고 유지 보수하는 단계에서 여러 문제점을 내포하게 된다. 대표적인 문제점으로 에이전트내 클래스들이 자신의 하드웨어나 운영체제 등의 플랫폼에 종속성을 갖고 구현, 유지 보수되어지는 관계로 다수의 에이전트 및 매니저들은 동일한 기능을 수행하는 서로 다른 클래스들을 모두 중복하여 유지해야만 하며 이로 인하여 전체 망관리 입장에서는 동일한 기능을 수행하는 클래스의 버전을 관리하기가 용이하지가 않다는 점을 들 수 있다[5][6]. 이러한 문제들을 해결하기 위하여 매니저와 에이전트 등 분산객체 내의 소프트웨어 컴포넌트들을 컴포넌트웨어(componentware) 형태로 생성하여 플랫폼독립형 클래스저장소 (PICR : Platform Independent Class Repository)[5][6][7][8]에 유지시켜 필요시 이러한 컴포넌트웨어들을 분산객체로 동적 또는 정적으로 다운로딩하여 실행시키는 Farming 방법론[8][9]을 제안하였다. 이 방법론은 TMN 시스템 내에서 에이전트를 구성 시 주어진 역할을 수행하기 위하여 분산객체의 기능블록들을 프레임워크 리스트로 정의하고 이 프레임워크 리스트를 구성하고 있는 소프트웨어 컴포넌트들을 PICR로부터 이식하여 분산객체 프레임워크를 구성하고 에이전트의 실행에 필요로 하는 데이터 컴포넌트까지도 이식하여 전체 에이전트를 구성하는 것을 의미한다. 특히 OLB(On-demand Loading Block)는 PICR에서 에이전트로 요구에 의해(on-demand) 동적으로 다운로딩하여 수행이 이루어진다.

Farming 방법론은 지식표현 모델인 Farmer 모델에 기본을 두고 있다. Farmer 모델[8]은 시스템 개체구조(System Entity Structure)[7] 의 개념을 도입한 프레임 구조모델로서 지식표현 모델인 EA (Entity-Aspect) 모델[7]에 기본을 두고 있다. Farmer 모델의 주요 목적은 실제 디자인 하고자 하는 에이전트를 분석하여 에이전트를 구성하고 있는 컴포넌트 요소들을 측면에 따라 분리, 추출해 내는 데 있다.

본 논문에서는 Farming 방법론을 이용하여 실제로 PCN (Personal Communication Network)의 TMN 에이전트를 구현시 나타나는 결과를 ATM과 같은 고속망을 DCN(Data Communication Network)으로 하는 경우에 대하여 성능평가를 하였다.

II. Farming 방법론

Farming 방법론은 데이터마이닝과 데이터 시스템 통합에서 다루고 있는 메타데이터 저장소의 구축과 유사한 개념으로 이해할 수 있다. 단지 Farming 방법론은 Farming 이라는 단어의 의미와 같이 저장소에 저장된 메타 소프트웨어 컴포넌트들을 실제 세상의 시스템을 구축하기 위해 구축하고자 하는 시스템의 프레임워크의 구조에 맞추어 메타 소프트웨어 컴포넌트 저장소로부터 아웃소싱하는 것이다.

ATM, FDDI 등 고속망을 기본망으로 사용하여 망관리 시스템을 구성할 경우 망관리 시스템내의 에이전트를 구성하는 소프트웨어 블록들을 컴포넌트웨어 형태로 변형한 후 기본망 내에 위치하는 임의의 서버로 이동시킨다. 이러한 서버를 클래스저장소라 정의하며 여기에 저장되어 유지되는 컴포넌트웨어들은 TMN내의 매니저 및 에이전트를 구성하는 하드웨어나 운영체제 등과 독립적으로 유지할 수 있기 때문에 플랫폼독립형 클래스저장소 (PICR : Platform Independent Class Repository)라 부른다. 그림 1은 개인휴대통신망 (PCN : Personal Communication Network) TMN 시스템에서의 PICR의 예를 보여주고 있다. PCN TMN 에이전트들이 수행해야할 기능들을 기능측면, TMN 관리계층 측면, 프로토콜 측면으로 구분하여 각 측면에 속하는 기능들을 컴포넌트웨어 형태로 변형한 후 이들을 PCN TMN내의 PICR에 저장시킨다. 그림 1 내의 CRBP(Class Repository Broking Processor)는 분산객체 내에서 다른 분산객체로 부터의 요구를 수행하기 위해 PICR로부터 해당 기능의 컴포넌트웨어를 임포트하는 실질적 역할을 담당한다.

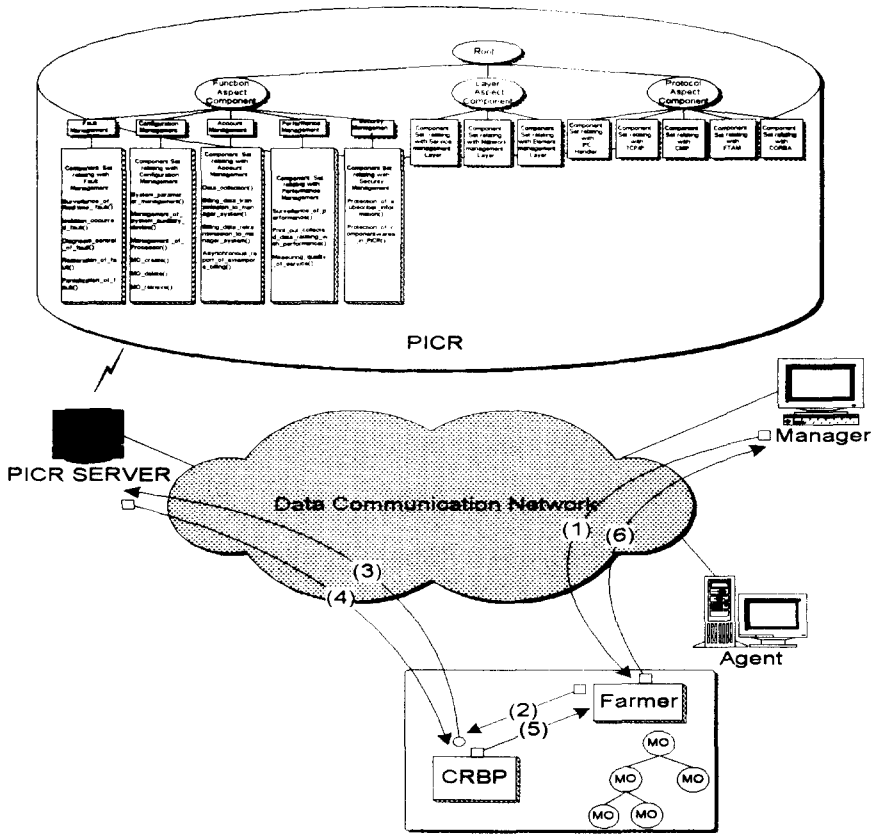

그림 1 클래스저장소를 갖는 분산객체형 TMN 구성모델

예를 들면 매니저가 신규 관리객체(MO : Managed Object)[1][2][3]를 생성할 목적으로 임의의 에이전트로 M_CREATE CMIP(Common Management Information Protocol) 메시지[1][2][3]를 보낸 경우, 에이전트내의 드라이버인 Farmer 모듈이 이 메시지를 수신한다. Farmer는 MO 생성 메시지를 CRBP(Class Repository Broking Processor)로 전송한다. 이 명령을 받은 CRBP는 MO생성과 관련된 컴포넌트웨어가 현재 자신의 에이전트내의 로컬 라이브러리에 존재하는 여부를 판단한다. 만일 존재하는 경우 해당 컴포넌트웨어가 로컬 라이브러리에 존재함을 알려 준다. 만일 로컬 라이브러리에 존재하지 않는 경우 가장 가까운 위치에 존재하는 PICR로부터 MO 생성과 관련된 컴포넌트웨어를 다운로드 받은 후 이를 Farmer로 넘겨준다. Farmer는 전송되어진 컴포넌트웨어를 수행함으로 MO를 생성한다. 이렇듯 실행에 필요한 소프트웨어 및 데이터 요소 컴포넌트웨어를 PICR로부터 에이전트로 다운로딩하여 수행하는 동작을 Farming 이라 한다.

TMN 망관리 에이전트를 설계시 전체 망관리 시스템의 구성과 관련된 형상 데이터, 즉 관리하고자 하는 분산객체 시스템 및 망 내의 노드 및 링크 수, 처리할 수 있는 트래픽의 한계치 등의 데이터 요소들은 분산객체 시스템의 구성시 소프트웨어 컴포넌트웨어와 더불어 에이전트 플랫폼으로 이식하는 것이 전체 망관리 시스템의 형상관리와 소프트웨어 버전관리라는 측면에서 바람직하다.

이와 같은 개념을 반영하기 위하여 소프트웨어 컴포넌트와 더불어 분산객체 시스템의 실행에 반드시 있어야 하는 데이터 요소까지도 플랫폼에 이식할 수 있는 방안으로 Farming 방법론을 제안하였다. 이 방법론에 의해 분산객체 망관리 시스템 내에서 분산객체를 구성시 주요기능을 담당하는 블록들을 프레임워크로 정의하고 이 프레임워크를 구성하고 있는 소프트웨어 컴포넌트 및 데이터 컴포넌트들을 PICR로부터 이식하여 에이전트를 구성한다. Farming 방법론은 다음과 같은 6 단계로 구분된다.

- 1 단계 : 네트워크 분석 및 관리대상 네트워크 요소의 선택
- 2 단계 : 분산객체 시스템의 분석 및 컴포넌트웨어 요소 추출
- 3 단계 : 메타 컴포넌트웨어 요소로의 변환 및 변환된 메타 컴포넌트웨어 요소의 PICR로의 이동
- 4 단계 : Farmer 모델 다이어그램에 의한 에이전트 프레임워크의 디자인
- 5 단계 : 측면객체 정의언어 (ADL : Aspect-Object Definition Language)로 코딩
- 6 단계 : 프레임워크의 구현

III. Farming 방법론의 평가

이번 장에서 Farming 방법론을 실제 TMN 에이전트의 구축에 적용(이하 Farming 방식)하였을 때 나타나는 결과에 대하여 Farming의 개념을 적용하지 않고 개발된 TMN 시스템과 비교하여 평가하였다. 기존의 TMN 에이전트는 DSET, RETIX 등의 에이전트 생성 툴킷[10]을 이용하여 개발되었으며 이러한 방안은 객체지향 프로그래밍 기법에 기본을 두고 있다. 하지만 이

방안을 이용할 경우에 에이전트를 구성하는 운영체제 및 하드웨어 등의 플랫폼에 종속적으로 에이전트를 개발해야 한다. 이 개발 방안을 Non-Farming 방법론이라고 본 논문에서는 부르기로 한다. 평가의 기준은 망에서 발생하는 CMIP 메시지, notification message(on-line message) 등과 같은 transaction 처리에 대한 응답시간을 주축으로한 성능으로 한다.

3.1 Transaction 응답시간에 따른 성능분석

CMIP 메시지 처리 및 에이전트에서 발생하는 notification 메시지 등을 처리하는 transaction t의 실행과 관련된 simulation 평가모델을 정의한다. 본 논문에서는 성능요소로서 각 분산객체 내에서의 CPU 처리시간, repository access 시간, 네트워크 요소들간의 통신시간 등을 고려하였으며 network topology는 고려하지 않았다. 우선 본 평가 모델에서 사용되는 용어들은 다음과 같이 정의된다.

s : transaction t (CMIP message, notification message 처리) 발생 노드
d : transaction t를 실제로 처리하는 노드 (에이전트, 매니저)
r : 에이전트가 관리하는 real resource
p : PICR server
N : site 집합 (s, d, p ∈ N)
T : transaction 집합 (t ∈ T)

본 simulation 의 모델은 그림 2와 같은 분산환경이 된다.

Transaction의 완료시간 CT_t 의 정의는 다음과 같다.
CT_t의 정의를 위하여 다음과 같은 용어들이 추가로 정의된다.

$CPU_T(x)$: 노드 x의 CPU 처리시간
$COM_T(s,d)$: 노드 s와 노드 d 사이의 통신시간
$COM_T(y,r)$: 노드 y(에이전트)와 real resource사이의 통신시간
ACC(p) : PICR access transaction 처리시간
$T_m(r)$: r 의 처리시간
$T_m(p)$: PICR server의 처리시간 (= $CPU_T(p)$ + ACC(p))
CT_{t_CMIP} : transaction t (CMIP 메시지)의 실행을 완료하는 데 소요되는 처리시간
$CT_{t\_notification}$: transaction t (notification message)의 실행을 완료하는 데 소요되는 처리시간
CT_t : transaction t 의 실행을 완료하는 데 소요되는 처리시간 (= CT_{t_cmip} + $CT_{t\_notification}$)

3.1.1 CT_{t_CMIP} 의 정의

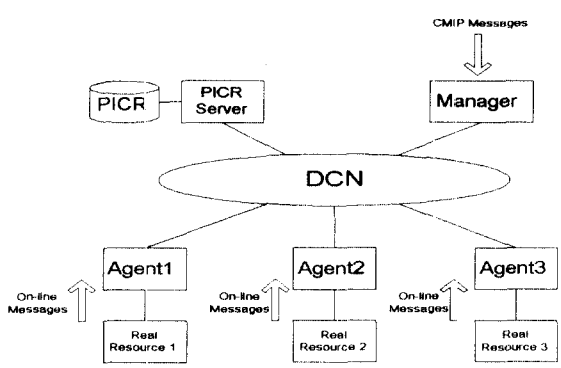

그림 2 Simulation 환경

발생 메시지 타입	빈도수 관련				
	발생장소	발생장소별 R_O	처리장소 별 R_C		발생메시지 갯수
CMIP	Manager	0.0	Agent 1	0.5	10^3
			Agent 2	0.3	
			Agent 3	0.2	
On-line	Agent 1	0.46	0.0		10^5
	Agent 2	0.29			
	Agent 3	0.25			

표 2 R_C 및 R_O

매개변수	manager	agent1	agent 2	agent 3	PICR server
$CPU_T(x)$	2.0	2.5	2.0	3.0	2.0
ACC(p)	0.0	0.0	0.0	0.0	1.0
Tm(r)	0.0	0.0	0.0	0.0	0.23

(단위 : ms)

표 3 실험에서 사용된 시스템 매개변수의 값

그림 2에서 매니저가 임의의 CMIP message를 TMN 내의 임의의 에이전트로 전송하는 경우 이를 처리완료하는 데 소요되는 시간으로 정의된다. CT_{t_CMIP} 는 매니저, 에이전트, PICR server와 같은 각 노드들의 CPU시간과 네트워크를 통한 노드들 사이의 전송시간, PICR disk 입출력 시간, 에이전트와 real resource 사이의 통신시간, real resource의 처리시간의 합으로 표현할 수 있다.

$$CT_{t\_CMIP} = CPU_T(s) + COM_T(s, d) + CPU_T(d) + COM_T(d, p) + CPU_T(p) + ACC(p) + CPU_T(p) + COM_T(p, d) + CPU_T(d) + COM_T(d, r) + Tm(r) + COM_T(d, r) + CPU_T(d) + COM_T(s, d) + CPU_T(s)$$

$$= \sum_{\alpha} CPU_T(s) + \sum_{\beta} COM_T(s, d) + \sum_{\gamma} CPU_T(d) + \sum_{\delta} COM_T(d, p) + \sum_{\varepsilon} CPU_T(p) + \sum_{\zeta} ACC(p) + \sum_{\eta} Tm(r) + \sum_{\theta} COM_T(d, r) \quad < 식\ 1 >$$

여기서 α에서 θ까지의 의미는 다음과 같다.
α : transaction t의 발생노드에서의 CPU 접근 횟수
β : s와 d 사이의 단위통신 경로를 통한 통신 횟수
γ : transaction t의 처리노드에서의 CPU 접근 횟수
δ : transaction t의 처리노드와 PICR server 사이의 통신 횟수 (*)
ε : PICR server에서의 CPU 접근 횟수 (*)
ζ : PICR disk 입출력 횟수 (*)
η : real resource에서의 처리시간
θ : real resource와 에이전트의 통신 횟수

위의 변수 중 *로 표시된 δ, ε, ζ 는 PICR내에 저장되는 ILB와 OLB의 비율에 따라 그 값이 결정된다. 즉, 에이전트내의

<식 1>	<식 2>	변수의 의미
$COM_T(d, r)$	$COM_T(r, s)$	real resouce와 에이전트와의 통신시간
$CPU_T(d)$	$CPU_T(s)$	에이전트에서의 CPU 처리시간
$CPU_T(s)$	$CPU_T(d)$	매니저에서의 CPU 처리시간
$COM_T(d, p)$	$COM_T(s, p)$, $COM_T(p, s)$	에이전트와 PICR server사이의 통신시간

표 1 <식 1>과 <식 2> 변수의 비교

framework에 ILB의 비율이 높을수록, OLB의 비율이 낮을수록 이들 변수의 값은 작아지며 그 반대의 경우는 이들 변수의 값은 커지게 된다. 따라서 이들 변수들은 simulation에 있어 중요한 역할을 담당하게 된다.

3.1.2 $CT_{t\_notification}$ 정의

TMN 에이전트들은 real resource에 fault가 발생하는 경우라든가 주기적인 상태보고 등과 같은 on-line notification message를 만들어 매니저로 보고를 한다. $CT_{t_notification}$은 이러한 on-lime message를 처리완료하는 데 소요되는 시간으로 정의된다. $CT_{t_notification}$는 매니저, 에이전트, PICR server와 같은 각 노드들의 CPU시간과 네트워크를 통한 노드들 사이의 전송시간, PICR disk 입출력 시간, 에이전트와 real resource 사이의 통신시간의 합으로 표현할 수 있다.

$$CT_{t\_notification} = COM_T(r, s) + CPU_T(s) + COM_T(s, p) + CPU_T(p) + ACC(p) + CPU_T(p) + COM_T(p, s) + CPU_T(s) + COM_T(s, d) + CPU_T(d) \quad <식\ 2>$$

여기서 <식 1>과 <식 2>의 일부 변수는 동일한 의미를 지니고 있으며 이는 표 1과 같다.

3.1.3 CT_t 의 정의

CT_t 는 transaction t 의 실행을 완료하는 데 소요되는 전체 처리시간을 의미하 며 다음과 같은 식이 성립된다.

$$CT_t = CT_{t\_cmip} + CT_{t\_notification} \quad <식\ 3>$$

표 1에 의하면 Tm(r) 변수만을 제외한 나머지 모든 변수들은 <식 1>과 <식 2>가 공유하고 있음을 알 수 있다.

3.1.4 실험 및 결과 분석

본 논문에서 제안한 Farming 방식을 이용하여 구현된 TMN 시스템의 simulation을 위해 Sun-Sparc 3에서 실행되는 SLAM II와 FORTRAN 77을 이용하였다. 적용대상으로 고려된 TMN 시스템은 그림 2와 같이 1 개의 매니저와 3 개의 에이전트 그리고 1 대의 PICR들을 연결하는 통신링크로 구성되어 있다. 본 적용대상에서는 매니저에서 발생하는 CMIP 메시지, real resource 1, 2, 3에서 발생하는 on-line 메시지 등이 있다. 본 실험에서 사용되는 에이전트 별 on-line 메시지 발생 빈도 R_O 와 매니저에서 발생된 CMIP 메시지의 에이전트별 처리 빈도 R_C 및 시스템 매개변수의 값은 표 2, 표 3과 같다. 본 수치들은 ATM TMN 시스템의 실측치로부터 얻었다. $COM_T(s, d)$와 같은 노드사이의 통신시간은 network dependent value로서 본 실험에서는 DCN을 Frame Relay(1.544 Mbps or 2.048 Mbps), FDDI or FDDI-II (100 Mbps), ATM(155 Mbps), Giga-Bit Net(2G)를 사용하는 경우로 조정해 가면서 시스템의 전체 성능을 비교하였다. 네트워크에서 통용되는 메시지의 평균 사이즈를 500K로 가정할 경우 각 프로토콜별로 다음과 같은 $COM_T(s, d)$ 값은 표 4와 같다. PICR과 PICR server 사이의 통신 시간인 $COM_T(p, s)$ 값은 1 ms로 정의하였다. 그림 3은 표 4에 나타난 대로 $COM_T(s, d)$의 매개변수 값의 변경하여 가며 실험한 결과 나타난 CMIP 메시지 처리 및 에이전트에서 발생한 on-line 메시지를 처리하는 전체 transaction의 평균 응답시간을 비교한 그래프이다. 실험결과 Farming 방식을 이용하는 경우 저속망에서는 non-Farming 방식과 많은 응답시간의 차이를 보이지만 고속망으로 갈수록 그 격차는 줄어드는 것을 볼 수 있다. 그림 4와 그림 5는 에이전트에서의 CMIP 메시지 처리 시의 평균 응답시간 및 on-line 메시지 처리 발생 시의 평균응답 시간을 보여주고 있다. 이들 그림 또한 그림 2에서와 같이 고속망으로 갈수록 Farming 방식과 non-Farming 방식의 격차가 줄어들음을 알 수 있다.

3.3 성능 및 개발 비용간의 관련성

DCN을 구성하는 망이 ATM 이상의 고속망으로 발전시 Farming 방식과 non-Farming 방식사이에는 성능에 있어 큰 차이를 보이지 않게 된다. 하지만 개발비용 면에서 본다면 Farming 방법론이 비교우위에 있게 된다. 따라서 100Mbps 이상의 고속망이 DCN으로 구성이 된다면 Farming 방법론에 의한 시스템의 개발이 바람직하다.

DCN	전송속도
Frame Relay	2,590 or 1,953
FDDI FDDI-II	40
ATM	25.0
Giga-Bit Net	2.0

(단위 : ms)

표 4 $COM_T(s, d)$의 매개변수 값

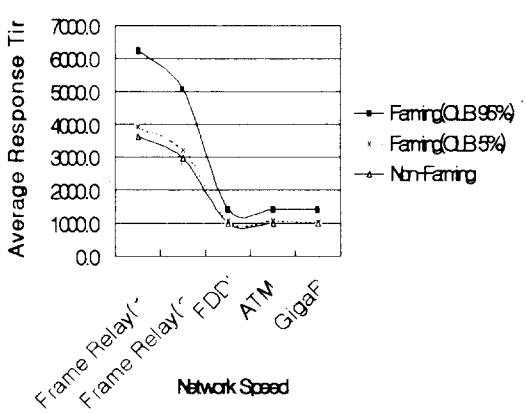

그림 3 전체 transaction 처리시의 평균 응답시간

IV. 결론

여러 통신망을 총괄적이고 효율적으로 운용하고자 출현한 TMN은 구축과정에서 서로 다른 하드웨어와 운영체제 등 상이한 플랫폼 환경 하에서 개발되는 관계로 매니저와 에이전트 등의 분산객체내의 클래스를 개발하고 유지 보수하는 데 여러 문제점을 내포하게 된다. 이러한 문제들을 해결하기 위하여 Farmer 모델에 기본을 둔 Farming 방법론이 제안되었다. Farming 방법론은 고속망을 기반으로 하는 클라이언트/서버, 매니저/에이전트 시스템의 설계 및 구현을 위하여 framework 및 소프트웨어 architecture의 개념을 반영한 설계 및 구현 방법론이다.

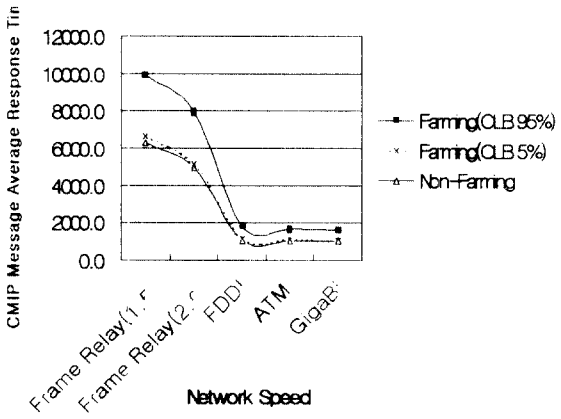

그림 4 CMIP 메시지 처리시의 평균 응답시간

구축된 시스템을 바탕으로 한 시뮬레이션 결과 DCN을 구성하는 망이 ATM 이상의 고속망으로 발전시 Farming 방식과 non-Farming 방식사이에는 성능에 있어 큰 차이를 보이지 않는다는 것을 볼 수 있었다. 하지만 개발비용 면에서 본다면 Farming 방법론이 비교우위에 있게 됨을 알 수 있었다. 따라서 100Mbps 이상의 고속망이 DCN으로 구성이 된다면 Farming 방법론에

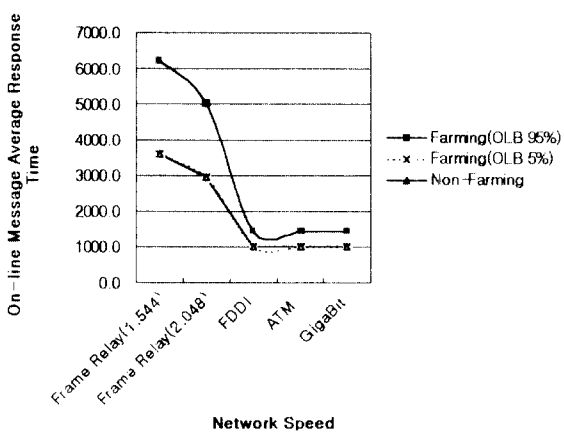

그림 5 On-line 메시지 처리시의 평균 응답시간

의한 시스템의 개발이 바람직하다는 결과를 얻을 수 있었다. 하지만 Farming 방법론은 실험에서 얻은 결과와 같이 non-Farming 방법론에 비하여 시스템의 성능이 적지만 저하된다는 단점을 가진다.

본 연구의 발전을 위하여 다음과 같은 분야에서 추후 연구가 진행되어야 한다고 생각된다. 첫째, Javabeans와 같은 플랫폼 독립형 인터프리터 언어를 구현에 이용할 경우 발생하는 시스템의 성능저하 문제를 우선적으로 해결해야만 한다. 둘째, 네트워크 내에서의 PICR의 위상과 관련된 문제를 해결해야만 한다.

V. References

[1] ITU-T Recommendation M.3010, "Principles for a TMN", 1992.

[2] ITU-T Recommendation M.3020, "TMN Interface Specification Methodology", 1992.

[3] ITU-T Recommendation M.3100, "Generic Network Information Model", 1992.

[4] ITU-T Recommendation M.3400, "TMN Management Function", 1992.

[5] Soo-Hyun Park, Sang-Hoon Park, and Doo-Kwon Baik, "15. Platform Independent TMN Componentware and Data Element Repository Based on Software Farming Methodology", *Systems Development Methods for the Next Century, Plenum Press*, pp.169 - 184, Edited by W.Gregory Wojtkowski, Wita Wojtkowski, Stanislaw Wrycza, and Joze Zupancic, 1997.

[6] Soo-Hyun Park, Doo-Kwon Baik, and Sang-Hoon Park, "Farming Methodology for TMN Platform Independent Class Repository Design", *In Proceedings of The 21st Annual International Computer Software & Applications Conference (COMPSAC'97), IEEE Computer Society*, pp.352 - 355, Washington D.C, USA, 1997.

[7] Soo-Hyun Park, Doo-Kwon Baik, and Sang-Hoon Park, "Platform Independent Class Repository on TMN in Personal Communication Network using Entity-Aspect Oriented Programming", *In Proceeding of The 4th International Workshop on Real-Time Computer Systems(RTCSA'97), IEEE Computer Society*, pp. 174 - 177, Taipei, Taiwan, 1997.

[8] Soo-Hyun Park, Sung-Gi Min, Doo-Kwon Baik, "Platform Independent TMN Agents Based on the Farming Methodology", *The IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, The Institute of Electronics, Information and Communication Engineers (IEICE), VOL.E81-A, NO 6, pp.1152 - 1163, Japan, June, 1998

[9] Soo-Hyun Park, Sang-Hoon Park, and Doo-Kwon Baik, "TMN PICR based on Software Farming Methodology", *In Proceedings of the KITE Summer Conference'97*, pp.121 - 124, Korea, 1997.

[10] DSET Corporation Version 1.2, "GDMO Agent Toolkit User Guide", 1995.