컴퓨터 네트워크 시뮬레이션을 위한 상이객체 지향 모델링

Various-Object-Oriented Modeling for Computer Network Simulation

오기은, 김명희*
Kee-Eun Oh, Myoung-Hee Kim

Abstract

본 연구는 새로운 객체 지향 모델링 방법으로 VOOM(Various Object-Oriented Modeling)을 제안한다. VOOM은 시스템의 요소 뿐만 아니라 시스템 요소들간의 관계 및 그들로 이루어진 구조를 서로 유형이 다른 프레임으로 정의하여 객체화시키고, 객체화된 시스템 요소 및 관계, 구조를 서로 연결하여 하나의 시스템을 구축하도록 한다. 따라서 전체 시스템의 측면에서 구성 요소의 역할과 전체적인 시스템의 행위 및 구조를 쉽게 파악할 수 있도록 한다.

VOOM을 통한 모델링 작업은 체계화되고 검증이 용이해지며, 모델의 유연성 및 재사용성이 확장되어질 수 있다. VOOM에 의한 모델링 작업 지원 가능성은 컴퓨터 통신망을 대상으로 한 시뮬레이션 모델링 예를 통해 제시하였다.

1. 서 론

대상 시스템의 복잡도가 높아지고 규모가 커질수록 그에 대한 모델링 작업은 점점 어려워진다. 예를 들면 대표적인 모델링 방법인 하향식 구조적 방법(top-down structured method)은 대형 시스템을 좀더 작고 간단한 서브시스템으로 나누고 그에 대한 모델링을 시도하는 분해(decomposition) 기법을 사용한다[7]. 이러한 방법으로 시뮬레이션 모델을 구축할 경우 대부분 직접적으로 세밀한 모델링(straightforward and detailed modeling)을 시도하게 되고, 따라서 이러한 방법의 사용은 많은 시간과 공간의 낭비를 가져올 뿐만 아니라 시스템 요소들을 자유롭게 변화시켜야 하는 여러 설계 대안들에 대한 모델의 구축 및 검증(validation)을 매우 힘들게 한다[2]. 또한 대형 시스템에 대한 모델링에 필수적인 모델의 재사용성(reusability) 및 확장성(extensibility) 부족의 문제점을 가져온다.

이러한 문제점들을 해결하기 위하여 객체 지향 모델링을 사용하려는 연구가 있어왔다. 그러나 기존의 연구들은 시스템 요소 객체화에 중점을 두고 있기 때문에 전체 시스템에서 요소가 다른 요소들에게 미치는 영향과 그들 사이의 관계 뿐만 아니라 전체 시스템 자체의 구조 및 행위를 파악하기가 힘든 문제점이 있고, 또한 시스템 구성 요소에 대한 객체만을 이용해 복잡한 시스템을 구성하도록 함으로써 모델링 작업을 더욱 어렵게 할 수 있다.

대형의 복잡한 시스템을 어떻게 체계적으로 간편하게 모델링하느냐 하는 문제는 시뮬레이션 모델 구축 뿐만 아

* 이화여자대학교 전자계산학과

니라 여러 응용 분야에서도 문제가 되는 기본 사항이라고 볼 수 있다. 본 연구에서는 새로운 객체 지향 모델링 방법으로 VOOM(Various Object-Oriented Modeling)을 제안한다.

본 연구에서 사용한 객체 지향 모델링 방법은 시스템의 요소에 대한 객체 지향에서 한 걸음 더 나아가 시스템의 요소 뿐만 아니라 시스템 요소들간의 관계 및 구조를 모두 객체화시킨다. 즉, 시스템 요소 및 요소들의 관계, 시스템의 구조를 각각 서로 유형이 다른 객체로 지정하고 이들을 서로 연결하여 하나의 시스템을 구축하도록 한다.

본 논문의 구성은 I장의 서론에 이어 II장에서는 일반적인 객체 지향 모델링 방법과 그 문제점에 대해 살펴보고, III장에서는 본 연구에서 제안한 새로운 객체 지향 모델링 방법인 VOOM에 대해 설명한다. IV장에서는 VOOM에 의한 컴퓨터 통신망 모델링 예를 보이고 마지막 V장에서는 결론을 내린다.

2. 기존 객체 지향 모델링 방법

대형의 복잡한 시스템의 모델링은 상호 복잡하게 얽혀 있는 구성 요소들과 그들간의 관계 및 상호작용(interaction)에 대한 올바른 정의를 필요로 하며, 현재는 구조적 방법 등과 같은 일반적인 모델링 방법들과 이들이 가지는 한계점의 극복을 시도하고 있는 객체 지향 모델링 방법을 사용하고 있다.

객체 지향 모델링은 크고 복잡한 시스템을 자신의 구조(structure)와 행위(behavior)에 따라 구성될 수 있도록 하는 프레임워크(framework)로, 크게 구조 모델(structural model)과 행위 모델(behavioral model)로 구성된다. 구조 모델은 실세계에서의 개체(entity)들에 대응되는 객체(object)와 그들의 관계에서 살펴본 한 시스템의 정적인 구조에 해당하는 모델로서 시스템내의 객체들과 그들의 특성, 그리고 그들간의 관계(relationship)를 정의한다. 행위 모델은 사건과 상태의 관점에서 시스템의 행위를 묘사해 놓은 동적 모델(dynamic model)로서, 시스템내에서의 객체들의 행위 뿐만 아니라 객체들간의 상호작용(interaction)을 정의하며 이를 통해 시간에 따라 변화하는 시스템의 여러 측면을 설명할 수 있는 모델이다. 객체 지향 모델링의 대표적인 특성으로 추상화, 캡슐화, 유형화, 상속성을 들 수 있으며, 일반적인 모델링 방법들에 비해 다음과 같은 장점들을 지닌다.

첫째, 객체 지향 방법은 추상화와 유형화 등을 이용해 객체 모델 및 행위 모델을 원하는 수준의 세밀한 부분까지 표현할 수 있다[11].

둘째, 객체 지향 방법을 사용함으로써 시스템 요소와 그들의 동적인 행동을 나타내는 데 있어 자연스럽고 강력한 파라다임을 제공받는다. 즉, 대상 시스템 요소와 모델링된 객체 사이의 일대일 대응관계(one-to-one correspondence)를 통해 객체화된 시스템 요소가 대상 시스템 요소의 행동을 올바르게 묘사하고 있는지 간단하게 검증할 수 있다[1].

셋째, 모델의 재사용성과 확장성이 뛰어나다. 객체 지향 모델은 하나의 객체내로 자료와 오퍼레이션을 캡슐화하여 모듈화되며, 계층적이고 상속성의 특성이 있으므로 변경이나 기능의 추가가 필요할 때 쉽게 구현될 수 있다[11][12].

그러나, 앞서 설명한 일반적인 객체 지향 모델링 방법은 다음과 같은 한계점을 갖는다.

첫째, 시스템 요소 객체화에 중점을 두고 있기 때문에 전체 시스템 측면에서 시스템 자체의 구조 및 행위를 체계적으로 이해하기 힘들다. 또한 전체 시스템에서 시스템 구성 요소의 역할 뿐만 아니라 요소가 다른 요소에게 미치는 영향 등을 정확히 파악하기 힘들다.

둘째, 시스템 요소 객체화를 전체적인 시스템 측면에서 관계 및 구조와 함께 고려하지 않기 때문에 실제로는 재사용성 및 확장성을 제한하게 된다.

셋째, 요소만을 객체화하고 시스템을 구축하기 위한 기타 사항들은 서술적인 방법이나 표기법(notation)을 사용하여 표현함으로써 기존의 다른 모델링 방법에 비해 효율적으로 모델링 작업을 간소화한다고 보기는 어렵다.

3. VOOM(Various Object-Oriented Modeling)

3.1 특징

시스템에 대한 관점은 크게 시스템을 구성하고 있는 요소들과 각 요소들간의 관계로 보는 정적인 관점과 시스템 기능들과 이를 위한 구성 요소들의 행위를 나타내는 동적인 관점으로 나눌 수 있다. 여기서 시스템 구성 요소란 시

스템을 구성하고 있는 물리적인 또는 개념적인 개체를 의미하고, 구성 요소간의 관계는 데이타를 바탕으로한 개체간의 정적인 구조를 의미한다.

본 연구에서는 앞서 언급한 정적, 동적인 관점을 바탕으로 시스템 구성 요소와 그들간의 관계, 구조를 정의한다. 자세히 설명하면 시스템에서 독립적인 기능을 수행하거나 의미를 갖는 물리적인 또는 개념적인 개체를 시스템 구성 요소로 정의하고, 구성 요소간의 관계는 시스템 또는 서브시스템의 기능을 수행하기 위해 이러한 요소들이 어떻게 상호작용 하는가를 나타내는 것으로 정의한다. 즉 동적인 측면에서 시스템 기능에 대한 구성 요소들간의 직접적인 또는 간접적인(의미상의) 상호작용을 관계로 정의한 것이다. 따라서 시스템의 전체 구조는 이러한 구성 요소들과 요소간의 상호작용을 나타내는 관계의 결합으로 정의되어질 수 있다.

본 연구에서 제안한 VOOM은 새로운 객체 지향 모델링 방법으로 크게 요소 객체화, 관계 객체화, 구조 객체화로 구분된다. 즉, 시스템의 요소 뿐만 아니라 시스템 요소들간의 관계 및 이들로 이루어진 구조를 모두 객체화시키고 이러한 요소, 관계 및 구조 객체를 서로 연결하여 하나의 시스템을 구축하도록 한다. 이러한 VOOM의 일반적인 특징은 다음과 같이 정리할 수 있다.

첫째, 시스템 요소 뿐만 아니라, 관계 및 구조를 객체화한다. 따라서 전체 시스템의 측면에서 요소의 역할과 전체적인 시스템의 행위 및 구조를 쉽게 파악할 수 있도록 한다.

둘째, 정의되어진 객체들을 연결하여 하나의 시스템을 구축하도록 함으로써 시스템에 대한 모델링 작업을 보다 체계화시키고 간소화시킨다.

표 3-1은 VOOM에서의 객체의 종류와 내용을 정리한 것이다. 본 연구에서는 VOOM을 이용한 객체 지향 모델링을 기반으로 하여 사건 중심 시뮬레이션 모델(event-oriented simulation model)을 구축하기 때문에 VOOM을 통해 정의되는 모든 객체들을 사건 중심의 관점에서 파악하도록 한다.

본 장에서는 요소 객체, 관계 객체, 구조 객체를 정의하는 각 프레임과 프레임을 구성하는 슬롯(slot)의 내용에 대해 상세히 설명한다.

〈표 3-1〉 VOOM에서의 객체의 종류 및 내용

객체 종류	내 용
요소 객체	시스템에서 독립적인 기능을 담당하는 요소를 객체의 특성과 동적인 행위로 정의
관계 객체	시뮬레이션 수행을 위해 요소 객체들이 어떻게 상호작용하는지를 명시하는 사건 전달(event passing) 관계 정의
구조 객체	요소 객체와 관계 객체로 이루어진 시스템 또는 서브시스템의 구조 정의

3.2 요소 객체

시스템에서 독립적인 기능을 담당하는 요소가 객체로 정의된다. 본 연구에서 요소 객체 정의는 대상 요소의 특성과 행위에 대한 정의를 통해 이루어지도록 하며, 정의된 요소 객체는 최종적으로 대상 시스템 시뮬레이션 수행의 한 부분을 담당하는 모듈로 변환되도록 한다. 앞에서 언급한 대로 본 연구는 사건 중심 시뮬레이션을 사용하므로 이 모듈은 사건 중심으로 정의되고 이를 위해 모듈 변환에 기반이 되는 객체의 행위를 사건 중심 관점에서 기술하도록 한다.

3.2.1 요소 객체 정의 프레임

요소 객체 정의 프레임의 형태는 다음과 같다.

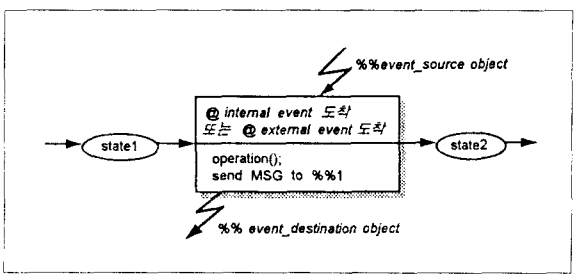
```
Name : 요소 객체의 이름
Parents : 요소 객체의 상위 클래스
Attributes : 요소 객체의 지역 변수
In_MSG : 외부에서 요소 객체로 들어오는 메세지
Out_MSG : 요소 객체에서 외부로 나가는 메세지
Behavior : 요소 객체의 행동
```

Name은 요소 객체의 이름을, Parents는 요소 객체가 속하는 상위 클래스의 이름을 나타낸다. 클래스들은 상속 계층구조(inheritance hierarchy)를 구성한다. 즉, 하위 수준에서 정의된 클래스는 상위 수준에서 정의된 클래스의 모든 특성을 상속받기 때문에 새로운 객체가 미리 정의된 클래스를 이용하여 쉽게 생성될 수 있도록 하는 확장성을 제공한다. Attributes는 요소 객체의 특성을 설명하는 지역변수들로 자신이 사용하는 자원(resource)들을 지정할 수 있다. In_MSG는 객체에 들어오는 메세지를 의미하는데 시

뮬레이션의 측면에서 보면 외부 객체로부터 도착한 외부 사건과 함께 전달된 메세지들을 나타낸다. Out_MSG도 In_MSG와 비슷하게 객체에서 나가는 메세지를 의미하고 외부 객체의 상태 변화를 일으키는 사건과 함께 전달될 메세지들을 나타낸다. Behavior는 객체의 행동을 즉, 객체의 함수적 특성들을 나타낸다. 마지막 Behavior 슬롯은 모듈로의 변환에 기반이 되는 객체의 행위를 정의하기 때문에 보다 상세히 살펴보도록 한다.

3.2.2 Behavior 슬롯

객체의 행위는 객체가 제공하게 될 서비스와 수행하게 될 기능을 의미하는 것으로, 어떤 조건과 사건 발생하에서 객체가 무슨 행위를 취하게 되는가를 파악하는 것이 필요하다. 행위는 또다른 사건을 유발시킬 수 있고 객체를 생성 또는 파괴하기도 하며 메세지를 교환할 수도 있다. 객체는 상태를 갖고 이 상태는 시간에 따라 변화를 일으키는데, 상태란 객체의 전체 상황이나 전체흐름에서 단계 따위를 의미한다[3]. 본 연구에서는 객체의 행위를 나타내는 Behavior 슬롯을 그림 3-1과 같은 상태망(state net)으로 표현하도록 한다.



〈그림 3-1〉 요소 객체의 상태망

상태망에서 state1과 같은 타원은 시스템 요소의 상태를, 타원을 연결하는 사각형은 상태전이(state transition)를 나타낸다. state1에서 state2로의 상태전이와 같은 시스템의 상태전이는 시스템의 상태를 변화시키는 외부 사건(external event) 또는 내부 사건(internal event)이 도착하는 경우에 가능하고 이때 명시된 동작(operation)을 수행한다. 물론 이 객체에서도 지정된 표현을 사용하여 다른 객체의 상태를 변화시키는 사건을 발생시킬 수 있다.

event_source object는 해당 객체의 상태를 변화시키는 외부 사건을 발생시킨 객체를 나타내고, event_destination object는 해당 객체가 상태를 변화시킬 대상이 되는 객체를 나타낸다.

시뮬레이션을 위해 요소 객체의 행위를 표현하는 상태망은 다음과 같이 미리 지정된 표현을 사용한다.

- @ A from B : 메세지 A를 보내는 사건이 객체 B로부터 발생하여 해당 객체에 도착했을 때(또는 도착하는 사건이 발생했을 때)를 의미한다.
- send A to B : 객체 B에게 메세지 A를 보내는 사건을 발생시키는 것으로 사실상 객체 B에서 'A from 사건 발생노드'라는 사건이 발생했음을 의미한다.
- %%number : 객체가 사건을 받거나 보내는 event_source/destination object가 정해지지 않았음을 나타내는 것으로 number는 1부터 시작하도록 한다. number에 대응하는 객체는 관계 객체화를 통해 지정된다.

요소 객체화에 관해 고려해야 할 문제는 정의된 요소 객체들이 시뮬레이션 모델로 어떻게 변환될 수 있는가 하는 것이다. 왜냐하면 일반적인 객체 지향 모델링을 적용했을 때 모델링과 시뮬레이션 모델 구축에는 상당한 차이가 있기 때문이다. 본 연구는 이러한 단점을 보안하여 객체 요소의 행위를 나타내는 상태망이 이산 사건 시뮬레이션 모델로 직접적으로 변환될 수 있도록 이를 가상 시뮬레이션 코드(pseudo simulation code)로 표현하도록 하였다.

그림 3-1의 상태망은 state1에서 state2로의 변환이 시스템의 상태를 변화시키는 event_source object에 의한 사건 발생에 의해 가능하고 이때 operation을 수행하며, 또한 지정된 표현을 사용하여 event_destination object에게 사건을 발생시킨 행위를 나타내는데, 다음과 같은 가상 시뮬레이션 코드로 1 대 1 변환된다.

```
event MSG_from_event_source object :
        /* state1→ state2 */
        operations();
        send MSG to event_destination object;
break;
```

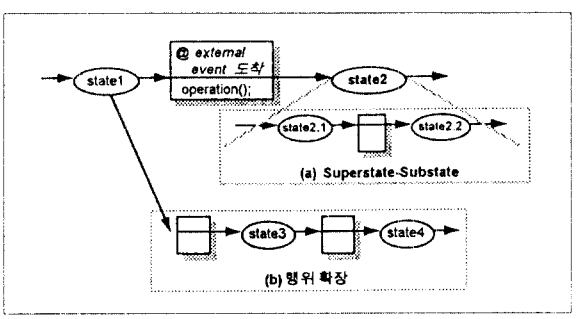
본 연구의 요소 객체화 특징은 객체의 행위를 그 객체와 함께 정의하는 것이다. 즉 객체의 구조적인 특성과 행위적 특성을 함께 객체화한다. 따라서 정의된 객체가 상

속성을 갖는다는 것은 변수들의 상속을 의미하는 구조적 특성의 상속뿐만 아니라 그 행위의 상속도 고려해야 한다. VOOM에서 행위의 상속은 아래와 같이 두 가지 방법으로 정의할 수 있다.

1) Superstate-Substate

Harel에 의해 제안된 Statecharts의 개념을 적용한 것으로[6] 상위 객체의 행위를 그에서 파생된 하위 객체에서 상세화하도록 정의한 것이다. 즉, 상위 객체의 상태를 superstate로 삼고, 하위 객체에서는 각 superstate들을 자신의 substate들과 substate들간의 변환들로 상세히 표현하는 것이다.

단, 외부사건의 발생과 같이 superstate로 들어오고 나가기 위한 상태 전이 조건을 하위 객체에서 변경시켜서는 안된다. 또한 임의의 시간에 객체는 superstate 중 오직 하나에만 머무를 수 있고, superstate내에서는 오직 하나의 substate에만 머무를 수 있다. 그림 3-2 (a)는 상위 객체의 state2를 superstate로 삼아 하위객체에서 state2의 상세화를 위해 state2.1과 state2.2, 그리고 그들간의 변환을 포함하도록 정의한 것이다. superstate-substate에 의한 행위 상속 모델링 예는 3.3.2절의 그림 3-3 (a)와 그림 3-4 (a)에서 찾아볼 수 있다



〈그림 3-2〉 행위 상속 예

2) 행위 확장(behavior extension)

상위 객체의 행위를 상속받은 하위 객체는 상위 객체의 상태나 변환을 지우거나 변경시킬 수 없다. 다만 새로운 state의 첨가, 변환의 첨가를 통해 상위 객체의 행위를 추가적으로 상세화시켜 자신의 행위를 표현한다. 그림 3-2 (b)는 기존의 상태망에 state3과 state4, 그리고 그들간의 변환을 첨가한 행위 확장 예이다.

3.3 관계 객체

관계란 요소 객체들 작동 결합(operational coupling)이다. VOOM에서 관계 객체화란 시스템에 대한 시뮬레이션 수행을 위해 객체들이 어떻게 상호작용(interaction) 하는가를 모델링한 것으로, 객체들간의 실제적인(물리적인) 상호작용 뿐만 아니라 시스템 기능에 있어서 요소 객체의 역할 등과 같은 의미적인 관계도 객체로 정의되어질 수 있다.

본 연구에서의 관계 객체는 객체들간의 사건 전달(event passing) 관계를 명시하는 것으로 정의한다. 관계 객체를 통해 일반적인 두 객체 사이의 관계뿐만 아니라 특정한 기능을 수행하는 셋 이상 되는 객체 사이의 관계도 객체화가 가능하다. 이러한 관계 객체화를 통하여 한 객체가 다른 객체들과 어떻게 상호작용을 하는지 쉽게 파악할 수 있는 장점을 얻게 된다.

3.3.1 관계 객체 정의 프레임

관계 객체 정의 프레임의 형태는 다음과 같다.

Name : 관계 객체의 이름 C_objects : 관계에 참여하는 요소 객체들 Activity : 요소간의 상호작용 Pass_MSG : Activity 수행시 전달되는 메세지

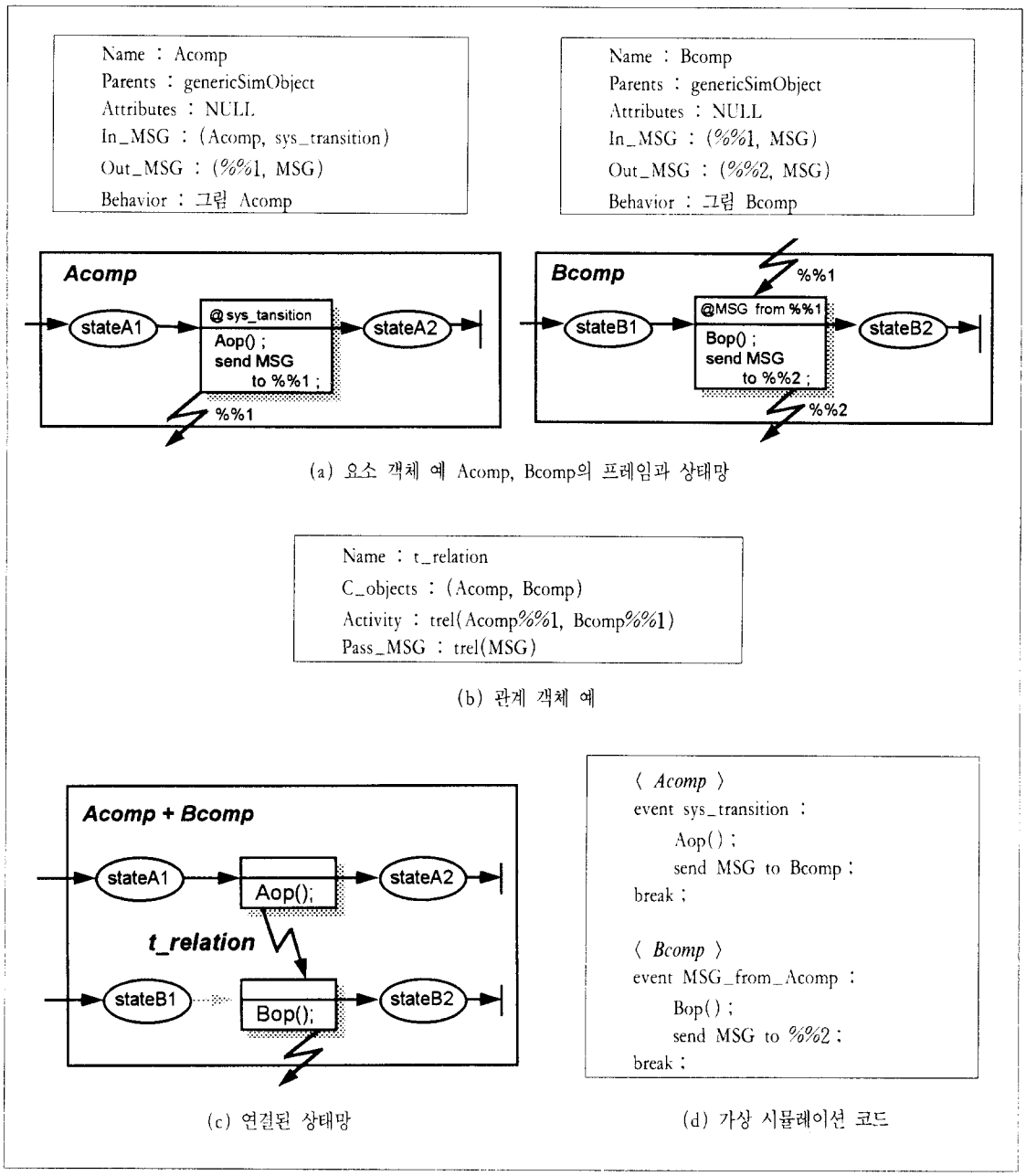
Name은 관계 객체의 이름을, C_objects는 관계에 참여하는 요소 객체들 즉, 상호작용이 있는 객체들을 명시한다. Activity는 관계에 참여하는 요소 객체들의 상호작용을 정의한다. 일반적으로 객체간의 상호작용은 반드시 객체간의 메세지 전달(message passing)을 통해 가능하기 때문에 정의된 Activity와 함께 전달되는 메세지를 Pass_MSG를 통해 명시한다. VOOM에서는 관계에 참여하는 요소 객체의 수를 제한하지 않으므로 한 관계 객체가 여러 객체 사이의 상호작용을 의미할 수 있다.

실제로 객체간의 관계는 Activity 슬롯에 의해 결정된다. 다음은 Activity 슬롯의 내용과 그 의미에 대해 좀더 자세히 살펴보기로 한다.

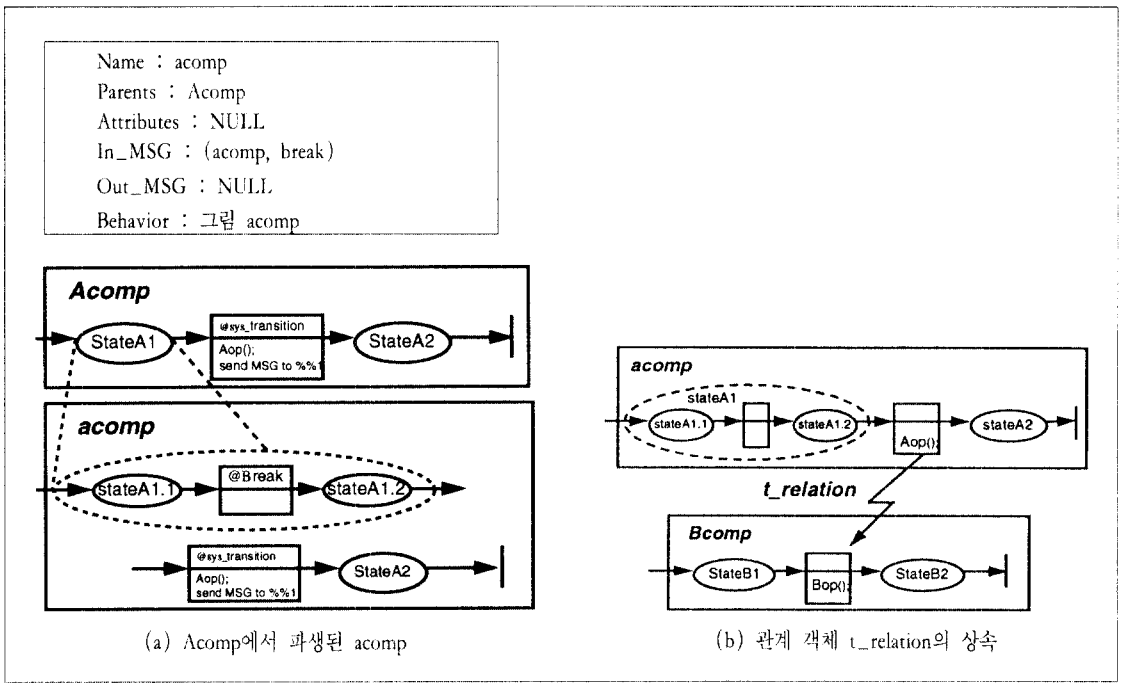
3.3.2 Activity 슬롯

Activity 슬롯은 상호작용하는 객체들과 실제로 이들이

어떻게 연결되는지 즉, 객체의 행위를 나타내는 상태들 중 어떤 상태에서 서로 연관되는 지를 명시한다. 다시 말하면 Activity 슬롯에 의해 정의된 관계 객체는 상호작용하는 객체들의 상태 연결을 의미한다고 볼 수 있다. 따라서



〈그림 3-3〉 관계 객체의 상태 연결 예



(a) Acomp에서 파생된 acomp (b) 관계 객체 t_relation의 상속

〈그림 3-4〉 관계 객체의 상속 예

관계 객체를 통해 한 시스템의 상태 변환이 다른 시스템의 상태 변환에 어떻게 의존하는가를 명시함으로써 시스템 의존 관계를 명확하게 파악할 수 있다.

예를 들어 그림 3-3 (a)에서 나타난 요소 객체 Acomp, Bcomp의 상태망에 대해 그림 3-3 (b)와 같은 프레임을 사용하여 관계 객체 t_relation를 정의할 경우 그림 3-3 (c)와 같이 연결된 상태망을 갖게 된다. 이때 각 요소 객체에 대한 가상 시뮬레이션 코드는 그림 3-3 (d)와 같다.

그림 3-3 (c)를 살펴보면 요소 객체 Bcomp에서 일어나는 stateB1에서 stateB2로의 상태 변환이 요소 객체 Acomp에서 일어나는 stateA1에서 stateA2로의 상태 변환에 의존하는 것을 알 수 있다.

관계 객체의 상태 연결 측면에서 상속문제도 함께 고려해 보아야 한다. 본 연구에서는 기본적으로 상위 요소 객체에 대해 정의된 모든 관계 객체는 그들로부터 파생된 모든 하위 요소 객체에 상속되도록 정의한다. 즉, 하위 객체는 상위 객체의 관계를 전혀 지울 수 없도록 하는 것이다. 그림 3-4 (a)는 그림 3-3의 Acomp에서 파생된 acomp를 나타내며, 그림 3-4 (b)는 Acomp와 Bcomp사이의 관계 객체 t_relation이 acomp와 Bcomp사이에도 그대로 존재함을 보여준다.

또한, 상세화된 하위 요소 객체는 그들 나름대로의 새로운 관계를 발생시킬 수 있으므로 하위 요소 객체에서 상위 요소 객체에 존재하지 않았던 새로운 관계를 정의할 수 있도록 한다. 이때 주의할 점은 이미 관계에 참여하고 있는 다른 요소 객체 또는 그 객체의 상위 요소 객체와 기존에 정의된 것과 같은 관계를 새로운 관계로 정의할 수 없다는 것이다.

따라서 새로운 관계의 정의는 다음과 같은 두가지 방법에 의해서 가능하다. 첫째, 전혀 새로운 요소 객체와의 관계를 정의한다. 둘째, 기존의 관계에 참여한 요소 객체와 전혀 다른 의미의 새로운 관계를 정의한다[6].

3.4 구조 객체

요소 객체와 관계 객체들로 구성되어지는 시스템내의 단위 구조를 하나의 구조 객체로 정의한다. 구조 객체화는 간단한 객체로부터 복잡한 객체를 구성함으로써 시스

템 전체 구조를 이해하기 쉽게 만들 뿐만 아니라 객체들의 집합에 대해 동시적으로 함수를 수행시킬 수 있는 메카니즘을 제공한다.

구조 객체에 대해 수행되는 함수들은 그들의 모든 요소들로 전파될 수 있다. 예를 들면 PBX에 대한 poweroff 함수는 자동적으로 그의 모든 요소를 poweroff하게 한다. 물론 구조 객체의 모든 요소들에게 전파되지 않도록 할 수도 있다[6].

3.4.1 구조 객체 정의 프레임

구조 객체 정의 프레임의 형태는 다음과 같다.

Name : 구조 객체의 이름 Com_Objs : 구조 객체를 구성하는 객체 Rel_Objs : 구조 객체를 구성하는 관계 객체 Connect : 정의된 객체간의 연결

Name은 구조 객체의 이름, Com_Objs는 구조에 참여하는 객체, Rel_Objs는 관계 객체를 의미한다. 구조에 참여하는 객체는 요소 객체 뿐만 아니라 미리 정의된 다른 구조 객체일 수도 있다. Connect는 관계 객체를 통해 정의되지 않은 객체간의 연결을 의미하고, 구조 객체간의 상호작용을 새로운 관계 객체로 정의할 수도 있다.

객체화된 시스템은 전체적인 시스템을 구성하는 하나의 서브시스템이 되어 전체 시스템 구조 객체를 구성하는 또 하나의 객체가 될 수 있다. 따라서 어떻게 시스템 구성이 되었는가 하는 것은 구성 계층구조(construction hierarchy)로 표현한다.

결과적으로 본 연구에 따른 시스템 정의는 전체적인 측면에서 상속 계층구조와 구성 계층구조로 표현될 수 있다. 즉, 상속 계층구조는 시스템을 구성하고 있는 요소 객체들의 상속 관계를 나타낸 것이고, 구성 계층구조는 상속 계층구조에 나타난 객체들이 어떻게 시스템을 구성하고 있는가를 나타낸 것이다. 따라서 객체는 상속 계층구조에서 오직 한번 나타나지만, 구성 계층구조에서는 한번 이상 나타날 수 있다.

그림 3-5는 일반적인 통신망의 상속 계층구조와 구성 계층구조의 예이다. 객체 genericObject에서 processor 요소 객체와 프로토콜과 관련된 protocolEntity 요소 객체가 파생되었지만 실제로 processor객체는 protocolEntity로 구성되고, protocolEntity는 layer1, FDDI의 layer2, layer3 객체들로 구성됨을 알 수 있다.

4. VOOM에 의한 컴퓨터 통신망 모델링

본 장에서는 VOOM에 의한 컴퓨터 통신망 모델링 예를 보인다. 먼저 대상 통신망과 VOOM에 의해 정의된 객체에 대해 설명하고 최종적으로 만들어지는 시뮬레이션 모델을 살펴본다.

4.1 대상 통신망

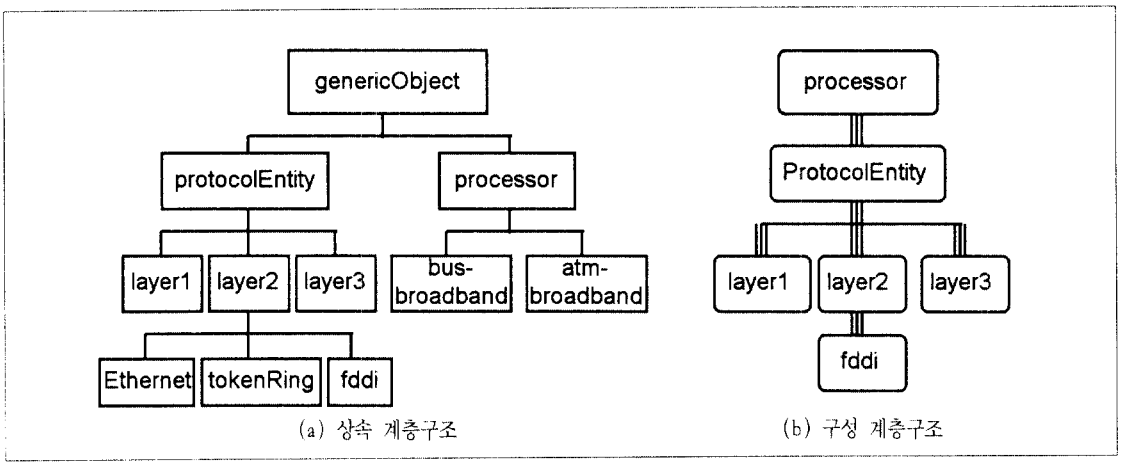
컴퓨터 통신망의 효율적인 설계 및 구축을 위해 통신망 시스템에 대한 성능평가는 반드시 필요한 작업이며 이를 위해 해석적 방법(analytic techniques), 시뮬레이션, 원형설계(prototype design)등을 사용하는데 모델의 유연성(flexibility)등의 이유로 시뮬레이션이 가장 많이 사용된다[3][4]. 그러나 컴퓨터 통신망이 점점 복잡해지고 대규모로 구축되어질수록 통신망에 대한 시뮬레이션 모델 구축은 점점 어렵게 된다.

기존의 컴퓨터 통신망 모델링 방법인 대기행렬 망 모델(Queueing Network Model), 페트리 네트 모델(Petri nets Model), 절차적 모델(Procedural Model)등을[5] 사용하여 이러한 컴퓨터 통신망에 대한 시뮬레이션 모델을 구축할 경우 모델의 재사용성 및 확장성 부족 등 앞에서 언급했던 문제들이 대두된다. 이러한 문제점들을 해결하기 위하여 본 연구는 VOOM을 이용한 새로운 객체 지향 모델링을 시도하고자 한다.

본 연구에서는 Ethernet을 사용하는 근거리 통신망을 대상 통신망으로 삼고 VOOM에 의한 모델링을 시도한다. 실제 통신망에 대한 메세지 도착 과정은 도착시간 간격 분포를 사용하고 메세지를 전송하는데 걸리는 시간은 서비스시간 분포를 사용한다. 통신망에 대한 토폴로지는 고려하지 않는다.

4.2 VOOM에 의한 객체 정의

본 절에서는 VOOM을 이용해 대상 통신망을 모델링하였을 때 정의되는 요소 객체, 관계 객체, 구조 객체에 대해서 설명한다.



〈그림 3-5〉 통신망의 계층구조 예

4.2.1 요소 객체

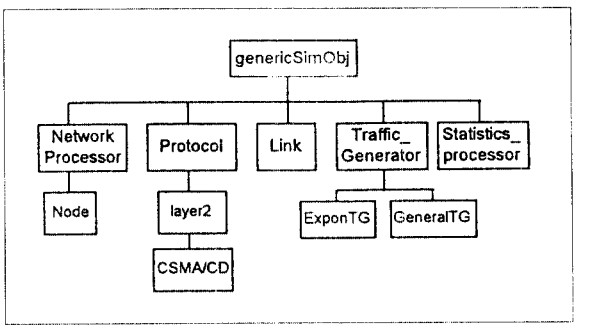
정의된 요소 객체들와 그 내용은 표 4-1과 같이 정리할 수 있다. 또한 요소 객체들은 그림 4-1과 같은 상속 계층 구조를 갖는다.

〈표 4-1〉 Ethernet 모델에서 정의된 요소 객체

요 소 객 체	내 용
NODE	사용자로부터 메세지를 받고 버퍼 같은 자원을 할당한 후 목적지로 메세지를 전송한다.
LINK	메세지를 보내거나 받기위해 사용하는 전송 설비를 나타낸다.
PROTOCOL	NODE는 정의된 프로토콜에 따라 LINK를 사용하여 메세지를 전송형태로 바꾸고 출발지에서 목적지까지 전송한다.
Traffic—Generator	사용자에 의해 정의되는 Traffic—Obj 객체에 따라 메세지를 생성시켜 NODE에게 보낸다.
Statistical—processor	메세지를 받아 원하는 통계량을 위해 처리한다. (예, 전체 통신망의 처리량, 평균 메세지 대기시간, 평균 전송 시도 횟수)
Traffic—Obj	전송될 메세지의 형태를 결정한다. (예, 메세지 형태(type), 메세지길이, 도착 시간 간격, 목적지)

정의된 요소 객체중에서 NODE, LINK, CSMA/CD에 대해 자세히 설명한다.

1) NODE : NODE는 사용자로부터 주어진 메세지를 정



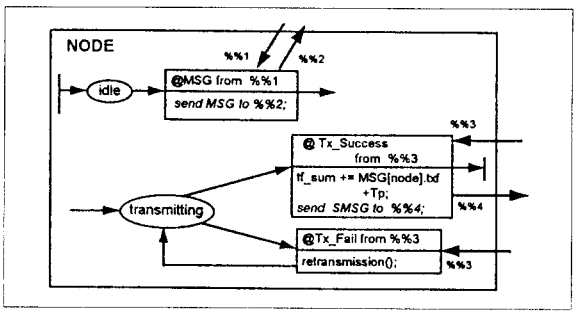
〈그림 4-1〉 Ethernet 모델에서 정의된 요소 객체의 상속 계층구조

해진 프로토콜을 사용해 전송하는 기능을 담당하는 객체이다. NODE 요소 객체는 아래와 같이 요소 프레임을 이용하여 정의되고 NODE의 기능 및 행위를 나타내는 상태망은 그림 4-2와 같다.

```
Name : NODE
Parents : genericSimObject
Attributes : ID, Buffer
In_MSG : (%%1,MSG) (%%3,Tx_Success)
         (%%3,Tx_Fail)
Out_MSG : (%%2,MSG) (%%4,MSG)
Behavior : 그림 4-2
```

외부로부터 MSG(메세지)가 도착하는 사건이 발생하면

idle상태에서 transmitting상태로 변환되고 이때 MSG를 %%2 객체에게 보낸다. 전송 성공의 사건이 도착하면 transmitting상태에서 ending상태로 변환되고, 통계 처리를 위해 MSG를 %%4 객체에게 보내 전송을 끝낸다. 전송 실패의 사건이 도착하면 계속 transmitting상태에 머물고 재전송을 시도한다.



〈그림 4-2〉 요소 객체 NODE의 상태망

앞서 설명한 바와 같이 NODE 요소 객체의 행위를 나타내는 상태망은 다음과 같은 가상 시뮬레이션 코드로 변환되어 저장된다.

```
event MSG_from_%%1 : /* idle → transmitting */
      send MSG to %%2;
break;
event Tx_Success_from_%%3 : /* transmitting → ending */
      tf_sum += MSG[node].txf + Tp;
      send MSG to %%4;
break;
event Tx_Fail_from_%%3 : /* transmitting → transmitting */
      retransmission();
break;
```

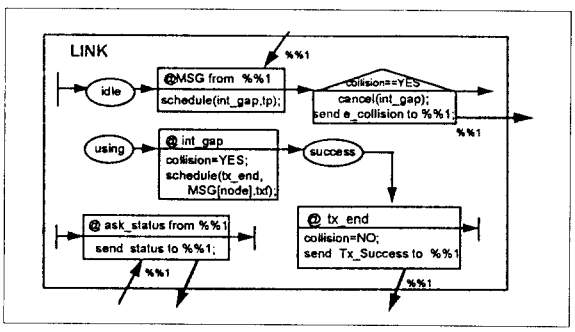
2) LINK : 메세지를 보내거나 받기 위해 사용하는 전송 설비를 나타내는 객체이다. LINK 요소 객체는 아래와 같은 요소 프레임을 이용하여 정의되고 행동을 나타내는 상태망은 그림 4-3과 같다.

전송할 MSG(메세지)가 도착하는 사건이 발생하면 idle 상태에서 나와 내부 사건 int_gap을 발생시키고 이미 LINK를 사용하고 있는 노드가 있는지 collision변수를 통하여 확인한다. 만약 LINK를 사용 중인 노드가 있으면

```
Name : LINK
Parents : genericSimObject
Attributes : collision
In_MSG : (%%1,MSG) (%%2,ask_status)
Out_MSG : (%%1,e_collision) (%%1,Tx_Success)
          (%%2,status)
Behavior : 그림 4-3
```

충돌(collision)의 발생을 알리는 사건을 발생시키고, 그렇지 않으면 전송 메세지의 길이에 따른 전송 시간이 지난 후 전송 성공의 사건을 발생시킨다.

다른 객체로부터 LINK의 상태를 묻는 사건이 발생하면 현재 상태를 되돌려 준다.



〈그림 4-3〉 요소 객체 LINK의 상태망

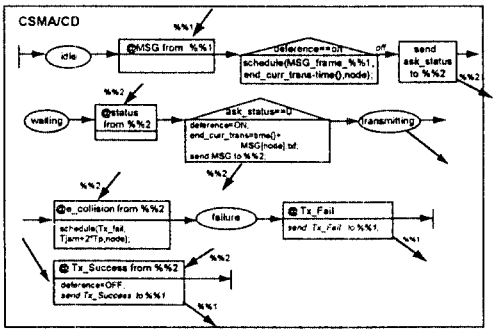
3) PROTOCOL : NODE는 정의된 프로토콜에 따라 LINK를 사용하여 메세지를 전송 형태로 바꾸고 출발지에서 목적지까지 전송하게 된다.

CSMA/CD(Carrier Sense Multiple Access With Collision Detection)는 둘 이상의 station이 하나의 공통적인 전송 매체를 공유하도록 하는 프로토콜이다. 그 알고리즘은 아래와 같다.

```
1. Check if medium is idle, transmit
2. If the medium is busy, defer until transmission is done
3. If collision, cease transmitting, transmit a brief jamming signal
4. Wait a random amount of time & retransmit
```

CSMA/CD는 아래와 같이 정의되고 상태망은 그림 4-4와 같다.

Name : CSMA_CD
Parents : PROTOCOL
Attributes : deference, end_cur_trans
In_MSG : (%%1,MSG) (%%2,collision)
(%%2,ask_status) (%%2,Tx_Success)
Out_MSG : (%%2,ask_status) (%%2, MSG)
(%%1,Tx_Fail) (%%1,Tx_Success)
Behavior : 그림 4-4



〈그림 4-4〉 요소 객체 CSMA/CD의 상태망

4.2.2 관계 객체

본 절에서는 NODE와 CSMA/CD의 관계 Transmit, CSMA/CD와 LINK사이의 관계 USE에 대해서 살펴본다.

1) NODE와 CSMA/CD의 관계 Transmit

NODE와 CSMA/CD의 관계 Transmit을 관계 프레임을 이용하여 다음과 같이 정의한다. NODE가 CSMA/CD에게 전송할 MSG를 보내고, CSMA/CD는 MSG 처리 후에 전송 성공/실패의 Tx_Success/Tx_Fail의 신호를 NODE에게 다시 보내는 관계를 Transmit으로 정의한다.

Name : Transmit
C_objects : (NODE, CSMA_CD)
Activity : Send(NODE%%2, CSMA_CD%%1),
Receive(NODE%%3, CSMA_CD%%1),
Pass_MSG : _Send(MSG),
_Receive(Tx_Success/Tx_Fail)

2) CSMA/CD과 LINK의 관계 USE

CSMA/CD과 LINK의 관계 USE를 관계 프레임을 이용하여 정의하면 다음과 같다. CSMA/CD가 LINK에게 전송할 MSG를 보내고, LINK가 충돌(collision)여부와 전송 성공 여부를 다시 CSMA/CD에게 보내는 관계를 USE로 정의한다.

Name : USE
C_objects : (CSMA_CD, LINK)
Activity : carry(CSMA_CD%%2, LINK%%1)
coll_check(CSMA_CD%%2, LINK%%1)
suc_check(CSMA_CD%%2, LINK%%1)
Pass_MSG : _carry(MSG) coll_check(e_collsion)
suc_check(Tx_Success)

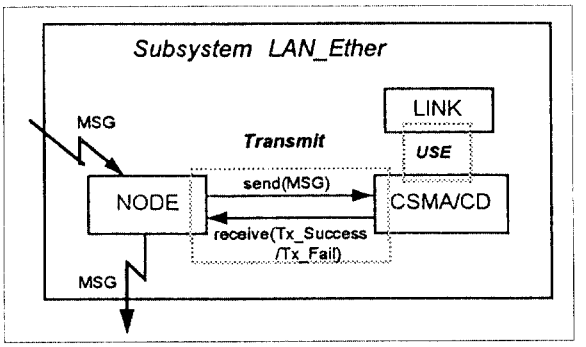
4.2.3 구조 객체

대상 통신망에 대한 구조 객체를 정의한다. 먼저 서브시스템으로 Ethernet에 대한 LAN_Ether을 정의하고 전체 근거리 통신망을 구성하는 Net_Sim을 정의한다.

1) LAN_Ether

LAN_Ether는 구조 프레임을 사용하여 아래와 같이 정의하고 이를 도식적으로 표현하면 그림 4-5와 같다. 위에서 정의한 NODE, LINK, CSMA/CD 요소 객체와 Transmit, USE관계 객체를 사용하여 통신망의 실제적인 부분을 담당하는 Ethernet에 대한 구조 객체를 만든다.

Name : LAN_Ether
Com_Objs : NODE, LINK, CSMA_CD
Rel_Objs : (NODE, LINK, Transmit),
(NODE, CSMA_CD, USE)
Connect : NULL

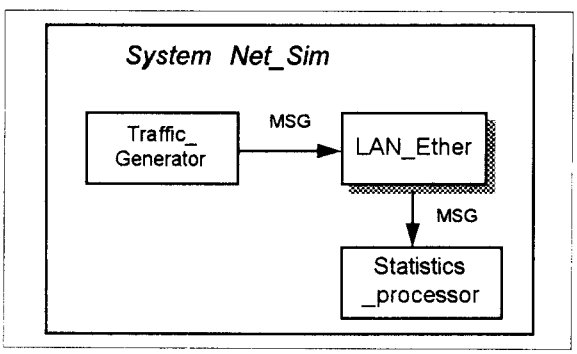


〈그림 4-5〉 서브시스템 LAN_Ether

2) Net_Sim

Ethernet을 사용하는 근거리 통신망은 Net_Sim 구조 객체로 아래와 같이 정의되고 그림 4-6과 같이 표현한다. 위에서 정의한 LAN_Ether 구조 객체와 메세지를 만들어 내는 Traffic_Generator 요소 객체, 메세지에 대한 통계량을 처리하는 Statistics_proc 요소 객체로 구성된다. Traffic_Generator에 의해 만들어지는 메세지는 Ethernet 모델인 LAN_Ether로 보내지고 메세지에 대한 처리(전송 성공 또는 전송 실패)가 끝나면 통계처리를 위해 Statisticsproc에 보내어 진다.

```
Name : Net_Sim
Com_Objs : LAN_Ether, Traffic_Generator,
           Statistics_proc
Rel_Objs : NULL
Connect : (LAN_Ether,NODE%%1,Traffic_Generator%%1)
          (LAN_Ether,NODE%%4,Statistics_proc%%1)
```

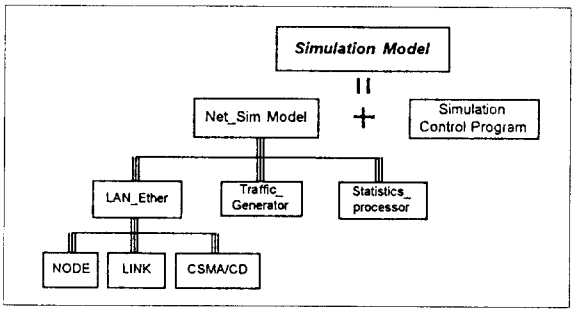


〈그림 4-6〉 시스템 Net_Sim

전체 시뮬레이션 모델은 Net_Sim에서 정의된 모든 요소, 관계, 구조 객체와 시뮬레이션 제어 모듈을 결합하여 만들어진다. 이를 구성 계층구조와 함께 나타낸 것이 그림 4-7이다.

4.3 시뮬레이션 모델

본 연구에서는 MacDoguall에 의해 개발된 사건 중심 시뮬레이션 언어인 smpl를 사용하여 시뮬레이션 환경을 구축한다. 즉, 앞에서 정의한 객체들로부터 smpl에 기반한 시뮬레이션 모델 즉, 시뮬레이션 프로그램을 만들어내는



〈그림 4-7〉 전체 시뮬레이션 모델의 구성

것이다.

사건 발생 루틴(event occurence routine), 사건 처리 루틴(event processing routine), 난수 발생 함수등과 같은 시뮬레이션 진행을 제어하는 프로그램들을 사용하기 위해 smpl을 사용했으며, 따라서 추상화 및 상속성과 같은 객체 지향의 특성을 지원하고 정의된 객체들로부터 시뮬레이션 프로그램을 만들기 위해 표 4-2와 같은 모듈들을 구성하였다. 시뮬레이션 시간 진행은 사건 발생루틴 내의 변수를 사용하여 제어하도록 하였다.

〈표 4-2〉 시뮬레이션 모델 구축을 위한 모듈

모 듈	기 능
모델 분석기	사용자가 정의한 모델을 지정된 입력 모델 구성 형식에 따라 분석하여 요소 객체, 관계 객체, 구조 객체에 대한 자료 구조를 생성한다.
모델베이스	기존에 정의된 객체들에 대한 정보를 유지 및 관리한다.
모델 생성기	모델 분석기와 모델베이스로부터의 정보를 이용해 정의되어진 컴퓨터 네트워크 모델에 대한 시뮬레이션 프로그램을 생성한다.

본 연구의 목적은 모델링 방법의 제안에 있으므로 시뮬레이션 프로그램으로의 변환 과정은 생략하고 본 절에서는 시뮬레이션 모델을 각 요소 객체에 따라 분리한 내용을 보이기로 한다. 그림 4-8은 LAN_Ether의 요소 객체 NODE에 대한 시뮬레이션 모델의 부분예이고, 그림 4-9는 요소 객체 CSMA/CD에 대한 시뮬레이션 모델의 부분예이다. 이를 살펴보면 각 부분 모델이 요소 객체의 행위 모델을 정의한 상태망을 거의 그대로 묘사하고 있음을 알

수 있다.

```
case MSG_from_Traffic_G :
    schedule(MSG_from_NODE, 0.0, node);
break;
case Tx_Success_from_CSMA_CD :
    tf_sum += MSG[node].txf+Tp;
    schedule(SMSG_from_NODE, 0.0, node);
break;
case Tx_Fail_from_CSMA_CD :
    retranmission(node);
break;
```

〈그림 4-8〉 요소 객체 NODE의 시뮬레이션 모델 부분예

```
case MSG_from_NODE :
            if (deference) /* ON */
            { schedule(MSG_from_NODE,
                    end_curr_trans-time(), node);
              break;
            }
            else
              schedule(ask_status_fro_LINK, 0.0,
              node);
break;
case status_from_LINK :
            if (status=YES)
            { deference=ON;
              end_curr_trans=MSG[node].txf+Tp+time
              ();
              schedule(MSG_from_LINK, 0.0, node);
            }
break;
case collision_from_LINK :
              schedule(Tx_Fail, Tjam+2*Tp, node);
break;
case Tx_Success_from_LINK :
            deference=OFF;
            schedule(Tx_Success_fro_CSMA_CD, 0.0,
            node);
break;
case Tx_Fail :
            schedule(Tx_Fail_from_CSMA_CD, 0.0,
            node);
break;
```

〈그림 4-9〉 요소 객체 CSMA/CD의 시뮬레이션 모델 부분예

5. 결 론

본 연구에서는 새로운 객체 지향 모델링 방법으로 VOOM을 제안하고, 제안된 VOOM을 이용하여 컴퓨터 통신망 시뮬레이션을 위한 모델링을 시도하였다.

VOOM은 시스템의 요소 뿐만 아니라 시스템 요소들간의 관계 및 그들로 이루어진 구조를 객체화시킴으로써, 시스템 구축을 위해 고려되어져야 할 여러 측면이 각각 객체화되어지고 이들의 연결로써 보다 체계적이고 용이한 대상 시스템에 대한 모델링이 가능하도록 한 확장된 객체 지향 모델링 방법이다. VOOM에 의한 모델링 작업 지원 가능성은 컴퓨터 통신망을 대상으로한 시뮬레이션 모델 구축 예와 해당 모델의 실행 가능성을 통해 제시하였다.

VOOM이 가지는 의의는 시스템의 여러 측면을 고려한 다양한 객체화를 통해 첫째, 대형의 복잡한 시스템에 대한 체계적인 모델링 작업의 지원이 용이해지고 둘째, 모델의 재사용성 및 확장성을 증가시키며 셋째, 모델 개발과 유지 보수를 위한 시간과 비용의 절감 효과를 얻을 수 있다는 것이다.

앞으로의 연구 과제는 무엇보다도 요소객체와 관계객체 사이의 semantic gap과 coupling completeness에 대한 문제 해결 등 다양한 일반 시스템의 모델링을 위한 VOOM의 보완 연구가 이루어져야 하며 또한 컴퓨터 통신망에 대해서는 ISDN과 같은 광역 통신망에 이르기까지 다양한 통신망에 대한 모델링의 가능성을 재시함으로써 실제로 VOOM의 활용을 시도하는데 있다.

참고문헌

[1] Adelsberger, H.H. et al, "Rule based object oriented simulation systems", Intelligent Simulation Environments, Simulation Series, Vol.17, pp.107-112, 1986

[2] Chlamtac, I. and R. Jain, "A methodology for building a simulation model for efficient design and performance analysis of local area network", Simulation, pp.57-66, Feb. 1984

[3] Edwards, G. and R. Sankar, "Modeling and simulation of networks using CSIM", Simulation, pp.131-136, Feb. 1992

[4] Frost, V.S., W.W. LaRue, and K.S. Shanmugan,

"Efficient Techniques for the Simulation of Computer Communications Networks", IEEE J. Select. Areas Comm., Vol.6, No.1, pp.146-157, Jan. 1988

[5] Kurose, J.F. and H.T. Mouftah, "Computer-Aided Modeling, Analysis, and Design of Communication Networks", IEEE J. Select. Areas Comm., Vol.6, No.1, pp.130-145, Jan. 1988

[6] Bapat, S., *Object-Oriented Networks: models for architecture, operations, and management*, Prentice Hall, Englewood Cliffs, 1994

[7] Booch, G., *Object-Oriented Analysis and Design with Applications*, 2nd, Benjamin Cummings, 1994

[8] MacDougall, M.H., *Simulating Computer Systems Techniques and Tools*, MIT, 1987

[9] Rumbaugh, J. et al, *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, 1991

[10] Zeigler, B.P., *Object-Oriented Simulation with Hierarchical, Modular Models*, Academic Press, San Diego, 1990

[11] 박성주 외, B-ISDN을 위한 객체 지향 시뮬레이션 모델링 기술 연구, 한국통신기술 주식회사 부설연구소-한국과학기술원 경영과학과 공동연구보고서, 1993

[12] 정보과학회지, 특집「객체지향 패러다임」, 한국정보과학회, 제11권, 제2호, 1993

● 저자소개 ●

김명희

1974.2 이화여대 사회학 학사
1979.8 서울대학교 전산학 석사
1986.2 독일 Gottingen 대학교 전산학 박사
1987.3~ 현 이화여자대학교 전자계산학과 부교수
관심분야 : 시뮬레이션 방법론 및 응용, 성능평가

오기은

1993.2 이화여대 전자계산학과 학사
1995.2 이화여대 전자계산학과 석사
1995.3~ 현재 삼성종합기술원 근무
관심분야 : 시뮬레이션 방법론 및 응용