技術解說

컴퓨터통신 프로토콜과 테스트 기법

崔 陽 熙
(한국전자통신연구소 망기술연구실장)

■ 차 례 ■

1 컴퓨터 통신 프로토콜

정보통신의 수요가 급증함에 따라 서로 떨어진 시스템사이의 정보전달을 수행하는 컴퓨터통신 프로토콜 또한 매우 빠른 발전을 하였다. 두 기기 사이의 간단한 데이터 통신을 제공하던 초기의 프로토콜으로부터 이제는 조직적이며 많은 고급기능을 수용하고 있으며 다수의 시스템간의 복잡한 통신양식을 모두 처리할 수 있는 망 architecture로 발전하여 왔다.

망 architecture의 개념은 IBM의 SNA (System Network Architecture)로부터 출발하여 여러 회사가 고유의 architecture와 이에 상응하는 프로토콜을 발표하기에 이르렀다[1]. DEC의 DNA (Distributed Network Architecture), Univac의 DCA (Distributed Communication Architecture), burrough의 network architecture, honeywell의 DSA (Distributed Systems Architecture)등 모든 컴퓨터 제조회사가 고유의 network architecture를 소개하였다.

최근의 추세는 어느 한 회사가 network architecture를 소개하는 추세에서, 여러 회사 또는 국가에서 공동으로 정의한 network architecture를 보급하려는 경향이 강하다. 그 예로서 미국의 연구기관, 대학 사이에서 발생한 ARPANET에 쓰이는 network architecture가 있고, 최근의 국제표준기구(ISO: International Organization for Standardization)에서 제창한 OSI (Open Systems Interconnection)이 있다[3].

이와같이 복잡한 발전은 다음과 같은 큰 문제점들을 야기시켰다.

P1. 서로 다른 network architecture는 서로 다른 컴퓨터 통신 프로토콜을 채택하므로 상호간의 직접접속이 불가능하다.

P2. 잘 알려진 network architecture나 프로토콜은 이를 처리하는 시스템이 불특정 다수의 회사에서 개발 공급되고 있다. 이 경우 같은 프로토콜을 모두 충실히 지키지 않는 경우, 같은 프로토콜을 구현한 제품끼리도 접속이 불가능할 수 있다.

첫번째 문제점(P1)의 해결책은 다음 두가지로 생각할 수 있다. 즉 모든 network architecture 또는 프로토콜마다 동일한 기능을 가진 다른 network architecture 또는 프로토콜로 변환시

키는 변환장치를 두어 이종 architecture 사이의 통신을 보장하는 것이 첫번째 해결책이다. 이러한 protocol conversion 접근 방식은 지금까지 널리 사용되어 왔다[2].

Protocol conversion의 주요 대상분야는 1) data link level의 프로토콜 사이(SDLCBSC, a-synchronous-synchronous 등), 2) LAN(Local Area Network)과 packet 망 사이, 3) 서로 다른 network architecture 사이(DNA-SNA, OSI-SNA, OSI-DCNA 등)로 나눌 수 있다. 대표적인 프로토콜, network architecture 사이의 protocol conversion 또는 gateway의 개발은 앞으로 장기간 계속될 전망이다.

P1에 대한 두번째 해결책은 지금이라도 통일된 규격의 프로토콜과 network architecture를 채택하여 모두가 사용하자는 것이다. ISO의 OSI가 가장 가능성이 큰 대안으로 보이며 이미 많은 컴퓨터 제조업자가(유럽의 13회사, 미국의 18회사가 이미 OSI 개발 보급을 공표하였다) OSI를 채택하겠다고 결정하였다. 이 방법은 지금까지 존재하는 모든 시스템을 protocol conversion의 방법으로 OSI에 정합시켜야 하는 단점을 갖고 있다. 그러나 이 방법은 여러 회사의 시스템을 모두 수용하여야 하는 응용분야에는 거의 필수적으로 고려되어야만 할 것이다. 예를들면 우리나라에서 근래에 추진하고 있는 국가기간전산망을 들 수 있다.

두번째 문제점(P2)이 발생하는 이유는 여러가지가 있다. 규격서(specification, standard 또는 recommendation) 자체가 모호한 부분을 담고 있어서 이를 구현할 때에 제품마다 다를수가 있거나, 규격서가 너무 복잡하게 쓰여져 있어서 구현하는 사람이 내용을 잘못 파악할 때 제품마다 다른 양식의 통신을 할 수 있다. 물론 규격서 자체가 오류를 내포할 때도 종종 있다.

P2를 해결하려면 프로토콜 규격서가 정확히 오류가 없도록 작성되어야 하며, 또한 구현된 제품이 과연 규격을 제대로 따르고 있는가를 test 할 수 있어야 한다. 즉 프로토콜 규격을 오해가 없는 formal description technique를 써서 기술하여야 하며, 이를 기준으로 implement된 기기들은 인정받고 있는 protocol test tool에 의하여 정확성을 판정받은 후에 사용되어야 한다는 것이다.

위의 두가지 issue, 즉 프로토콜 묘사기법과 프로토콜 테스트기법이 본고의 주요 내용으로 다음 절에 자세히 설명되고 있다. 또한 한국전자통신연구소에서 구축중에 있는 Protocol Test Facility (PTF)에 대하여서도 소개하고자 한다.

2 프로토콜 묘사 기법

프로토콜 규격서가 포함해야 할 사항으로는 제어순서, 메세지 코우딩방법, 성능기준, 에러검출 및 처리방법, 상위 및 하위계층과의 인터페이스방법이다. 이를 묘사하는 규격서는 모호한 부분을 담고 있어서는 안되며, 가능한한 간결하게 묘사되어야 하고 중복성이나 over-specification을 피해야 한다. 또한 규격작성방법이 기계적인 방법에 의한 규격자체의 검증을 가능하게 하면 좋다. 즉 automatic 또는 semi-automatic 방법에 의하여 규격의 오류 유무의 검증, 규격을 implement, 그리고 규격을 구현한 제품을 test하는 test scenario의 제작등을 할 수 있는 Formal Description Technique (FDT)가 권고된다.

FDT는 state개념에 기초를 둔 부류와 event에 기초를 둔 부류로 크게 나눌 수 있다. CCITT의 SDL (Specification and Description Language)[7], Petri Net[6], 그리고 ISO의 Estelle[5] 등은 state를 바탕으로 하였고, ISO의 LOTOS[4], Temporal Logic 등은 event에 기초를 두었다. 어느 방법이나 protocol module이 동작하는 내용을 묘사하는 데에 있어 방법상의 차이일 뿐 내용상의 차이를 보이지는 않는다.

State에 기초를 둔 것중 가장 간단한 것이 FSM (Finite State Machine)이다. 한 module이 가질 수 있는 모습을 한정된 갯수의 state로 mapping 할 수 있다고 보며, protocol은 어떤 module이 가능한 state 사이를 어떻게 움직이는 가를 묘사함으로써 파악된다는 개념이다. state transition은 외부의 자극에 의해 일어나며 module은 정해진 행동(action)을 취한다음 다음 정해진 state

로 이동하게 된다.

예를 들어 call establishment의 과정을 살펴보자. Module은 "idle"이라는 state에 있다가 상대편 system의 module로 부터 call request라는 메세지를 수신한다. Call request라는 자극은 수신 module로 하여금 call accepted라는 메세지의 송신(이것이 action이다)을 행하게 하고 "established"라는 다음 state로 넘어가게 한다.

이러한 FSM은 복잡한 프로토콜을 묘사하고자 할 때 state의 갯수가 너무 많아져 이용할 수 없다. 따라서 extended finite state machine 이 나타났는데, 이 경우 한 module의 state space는 여러개의 variable에 의해 결정된다. 이 variable 중 하나가 "STATE"라 불리우며 FSM의 state에 대응한다. 다른 variable의 예로서 message sequence number, buffer 사용율, timer등이 있다. "STATE"사이의 transition이 일어나려면 각 variable들이 주어진 조건을 만족해야 하며, transition이 일어남에 따라 variable의 값도 변동하게 된다.

Estelle이 extended FSM의 대표적 예이며 Pascal 언어와 유사한 구조를 갖고 있다. 그림 1에 표시된 transition은 ESTAB이라는 "STATE"에서 N. data response라는 외부로부터의 message를 받으면, 수신module에서 받은 message의 type이 data인가 확인해 보고 맞는 경우 begin과 end사이에 표시된 action을 취한 다음 다시 ESTAB로 돌아오는 것을 표시한다. 그림 1은 alternating bit protocol의 일부분에 해당한다.

프로토콜이 parallel process의 집합이라는데 착안하여, parallel process의 묘사에 쓰이는 Petri Net 기법을 쓰고 있다. 한 module의 내부상태는 여러 종류의 condition중 어느 것이 현재만족되고 있는가를 표시함으로서 묘사되고, state transition은 만족된 condition의 종류가 변화함으로서 정의된다. 그림 2에서 circle은 condition을 나타내며 현재 만족된 condition은 circle안에token (그림에서 ● 으로 표시됨)을 갖게 된다. circle과 circle들 사이는 transition을 나타내는 bar가

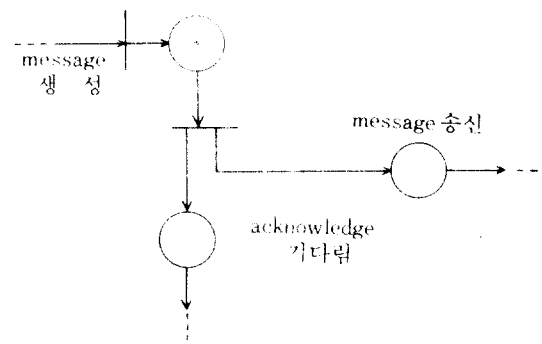


그림 2 Petri Net의 실제 응용예

```
FROM ESTAB TO ESTAB
WHEN N.DATA response
PROVIDED N. DATA. id = DATA
begin
        q. msgdata := NData. data;
        q. msgseq := NData. seq;
        Send. ack (q)
        if NData. seq = recv. seq then
        begin
          store (recv. buffer, q);
          incr-recv-seq
        end
end;
```

그림 1 Estelle의 실제 사용예

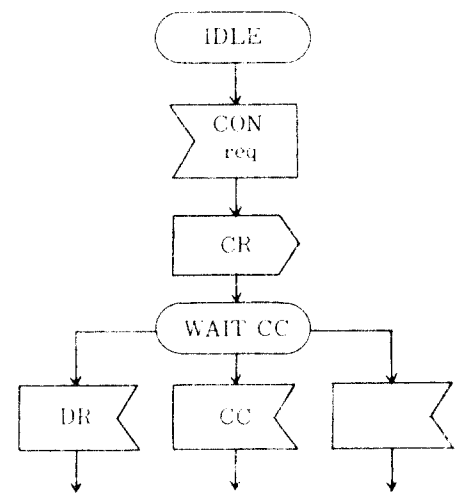


그림 3 SDL의 실제사용 예

있으며 transition은 bar를 향하고 있는 모든 circle이 모두 token을 가지고 있는 경우에만 가능하며, transition이 끝나면 bar를 향하고 있는 모든 circle의 token은 bar에서 나가는 쪽에 달린 circle로 이동하게 된다. 물론 FSM과 마찬가지로 transition을 할때 여러 action이 수반된다.

SDL은 estelle과 기능상 거의 동일하나 묘사하는 notation이 다르다. SDL은 programming language 같이 text로 표시하는 것과, 그림으로 표시하는 방법을 모두 제공하고 있다. 그림3에 간단한 예를 보였다.

LOTOS는 estelle과 함께 ISO에서 표준화를 추진하는 기법으로서 module의 성격을 module과 module간의 interaction point에서 일어나는 event들의 시간적인 인과관계를 정의해 줌으로서 완벽히 표현해 주고 있다. Temporal ordering primitive들이 쓰이며, 이들은 높은 abstraction을 지니므로 간결하게 규격을 작성하는데 도움을 준다.

FDT의 종류는 위에 소개한 이외에도 많은 종류가 있으나 일반적으로 특별히 우수한 성능을 보이는 것을 하나 고르기는 어렵다. 사용자에게 쓰이는 편리한 정도, 구현할 때 주는 도움의 정도, test의 용이성을 주로 판정 기준으로 한다면 SDL이나 Estelle이 가장 활발히 쓰이는 실정이다. 그러나 petri net이나 LOTOS도 빠른 발전을 하고 있다.

3 Protocol Testing

프로토콜 테스트의 중요성은 1절에서 설명한 바와 같다. 프로토콜 테스트는 hardware나 software로 구현된 제품이 본래의 규격을 얼마나 충실히 따르는 가를 정성적, 정량적인 면에서 검사하는 것으로 simulation이 아닌 real-time test를 수반한다. (protocol validation은 규격자체가 완벽한가를 주로 해석적, simulation 등을 통하여 검사하는 것으로 대개 구현하기 이전에 이루어진다).

컴퓨터 통신 프로토콜이 network architecture 속에 흡수됨에 따라 각 시스템은 여러개의 계층, 여러 종류의 프로토콜을 동시에 구현하게 된다. 이 경우 프로토콜 테스트는 일반적으로 계층별, 프로토콜별로 차례로 수행하게 된다. ISO의 OSI reference model의 notation을 따라 (N)-layer의 프로토콜을 검증하려고 할 때 다음 세가지방법 중 하나를 택할 수 있다[8]

a) local test method:

그림4에서 보듯이 (N)-entity under test의 상위 interface와 하위 interface에 test program을 직접 붙여서 (upper tester는 (N+1)-entity, Lower Tester는 (N−1)-entity를 simulation한다) (N)-entity의 동작을 직접 control 하고 관찰하는 방식이다.

b) distributed test method:

그림5에서 보듯이 상위계층 interface는 a)와

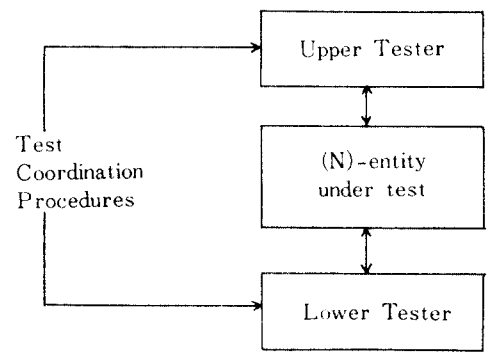


그림 4 local test method

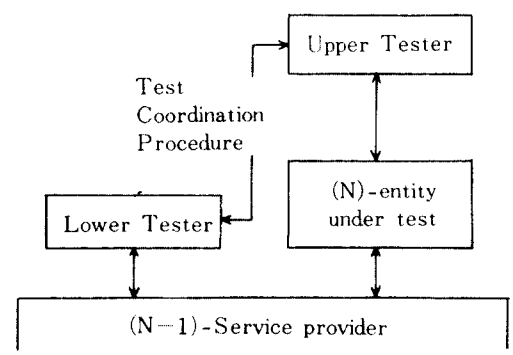


그림 5 Distributed test method

같이 직접 이용하나 하위계층 interface는 remote에서 control 하는 방식이다. 따라서 (N-1)-service는 이미 구축된 통신망을 이용한다.

c) remote test method:

b)와 동일하나 (N)-entity가 구현된 시스템내부에 upper tester를 install할 수 없는 경우를 말한다.

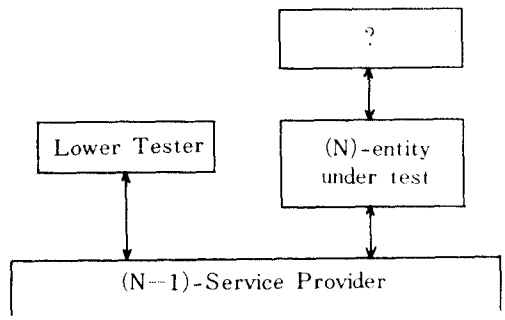


그림 6 Remote test method

위의 세 방법 모두 single layer의 test를 기준으로 소개하였으나 multi-layer testing으로 쉽게 확장할 수 있다.

현실적인 면에서 보면, remote test method가 가장 널리 쓰일 수 있다. 즉 이 방법은 test 대상 시스템 내부에 특별히 추가할 사항이 전혀 없기 때문이다. 반면에 다른 두 방법은 시스템 내부에 upper tester나 lower tester를 설치하여야 하는 부담을 안고 있다. 이것은 이미 개발 완료된 시스템에 적용하기 불가능하다. 따라서 기능적으로 가장 완벽한 test를 제공하는 distributed test method를 쓰려면 시험대상 시스템에 upper tester를 설치할 수 있도록 시스템 개발단계에 미리 요구하여야 한다. 이러한 추세는 프로토콜 테스트의 중요성이 대두된 이래 여러 나라에서 이미 채택하고 있다.

4 ETRI-PTF

프로토콜 테스트를 위한 검증시스템(protocol test facility)은 타 전자제품과 마찬가지로 공인된 기관에서 보유하며 형식승인의 절차에 따라 제품을 평가하는 것이 바람직하다. 현재 우리나라는 이러한 시스템을 보유하고 있지 않으며 한국전자통신연구소(ETRI)에서 개발연구가 진행중에 있다.

ETRI에서 개발중인 ETRI-PTF는 OSI model에서 data link 계층으로부터 application 계층까지의 주요 프로토콜을 검증할 수 있는 시스템으로서 distributed test method를 채택하고 있다. Lower layer (data link, network)를 위한 configuration과 Upper layer (transport, session, presentation, application)를 검증하기 위한 configuration을 따로 설명하기로 한다[9].

a) Lower Layer Testing: ETRI-PTF

ETRI-PTF는 DEC의 VAX 11/750 (UNIX 4.2 bsd)와 protocol simulator (HP4955A)로 구성되어 있다. VAX내에는 각종 test scenario를 위한 database와 parameter들이 저장되어 있다. Remote site에 위치한 system under test는 operator를 위한 terminal과 실제 구현된 SUT (System Under Test)로 구성된다.

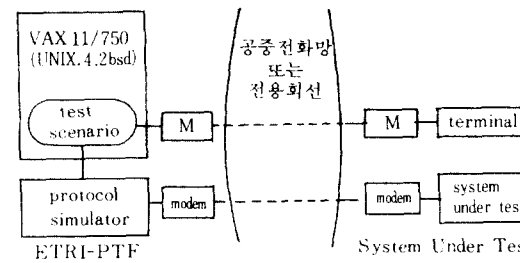


그림 7 ETRI-PTF: Lower Layer Testing

Test를 수행하기 위하여 remote에 있는 operator는 terminal을 통하여 VAX에 login을 한다. 그리고 test scenario를 선택하고 적당한 parameter들을 선택하게 된다. 이렇게 test할 종류를 고른 다음 test 수행을 terminal로부터 명령하면 test scenario들은 자동적으로 VAX로부터 protocol simulator로 download된 다음 protocol simulator와 SUT간의 test가 수행된다. test가 끝나면 protocol simulator로부터 test 결과가 VAX로 Upload되며 이 결과는 각종 분석 program을 통하여 operator가 원하는 형태로 정리

된 다음 operator의 terminal에 display 또는 print된다.

ETRI-PTF에서 대상으로 하는 프로토콜은 CCITT의X.25(link level, packet level), common channel signaling의 message transfer part, integrated service digital network의 user access protocols 등이 있다.

b) Upper Layer Testing: ETRI-PTF

Lower layer testing 경우와 달리 upper layer testing의 경우는 공중통신망으로서 packet 망을 직접 이용할 수 있다. distributed test method이므로 ETRI-PTF에는 upper tester에 해당하는 Test Driver가 있고 SUT에는 lower tester에 해당하는 test responder가 있다. Test를 위한 scenario는 앞의 lower layer testing의 경우와 마찬가지로 parameter, scenario database로부터 사용자의 선택에 따라 preprocessing을 거친다음 test driver- test Responder로 각각 입력된다.

ETRI-PTF는 (N)-entity의 표준구현제품인 reference implementation을 가지고 있다. RI가 처리할 수 없는 특수한 경우(예 : Reset 등)는RI의 아래에 위치한 exception generator에서 처리한다.

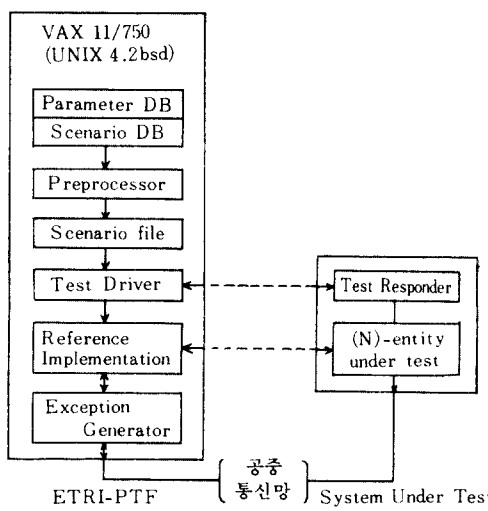


그림 8 ETRI-PTF: Upper Layer Testing

ETRI-PTF는 상위계층 프로토콜로서 ISO의 Transport, Session, CASE (Common Application Service Element), file transfer, message handling system을 일차대상으로 하며 필요에 따라 새로운 프로토콜을 쉽게 추가할 수 있도록 설계된다.

5 결 론

컴퓨터 통신 프로토콜이 복잡해지고 다양해짐에 따라 프로토콜의 정확한 묘사와 시험이 날로 문제시 되고 있다. 우리나라의 경우 지금까지 모두 외국의 몇 특정회사의 network architecture와 이에 따른 제품에 전적으로 의존하여 왔으나 종합정보 통신망이나 국가기간전산망과 같은 대형 프로젝트를 통하여 컴퓨터 통신 프로토콜의 자립화가 추진되고 있는 실정이다. 이 경우 프로토콜의 선택, 규격화, 구현 및 검증이 모두 매우 중요한 이슈로 등장할 것이므로 연구 개발이 시급하다고 볼 수 있다.

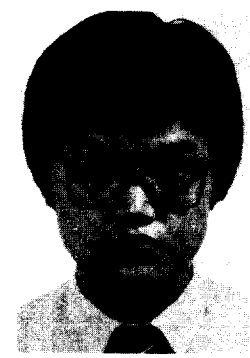
컴퓨터 통신 프로토콜의 연구는 Plan-Design-Specification-Validation-Implementation-Test-Operation (PDSVITO)으로 역시 일곱단계로 나누어지며 어느 것하나 소홀히 할 수 없다[10]. 본고에서는 이 중에서 Specification과 Test를 중점으로 다루었는데, alidation과 mplementation, design 등에 대해서도 국제적으로 매우 활발한 연구가 진행되고 있다. □

참 고 문 헌

1. A. Meijer and P. Peeters, Computer Network Architectures, Pitman Books, 1982.
2. P.E. Green, Jr., "Protocol Conversion," IEEE Trans. Communications, Vol. COM-34, no.3, pp.257-268, March 1986.
3. ISO, Information Processing Systems-Open Systems Interconnection - Basic Reference Model, IS7498.
4. ISO, Information Processing Systems- Open Systems Interconnection - LOTOS-A Form-

al Description Technique based on the temporal ordering of observational behavior, ISO TC97/DP8807.

5. ISO, Estelle-A Formal Description technique based on an Extended State Transition Model, ISO TC97/DP9074.
6. M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models," Computer Network, no.6, pp.419-441, June 1982.
7. CCITT, Recommendations Z.101-Z.104,Red Book, Generva, 1985.
8. Working Draft for OSI Conformance Testing methodology and Framework, ISO TC97/SC21 N410, Feb. 1985.
9. Yanghee Choi, "Trends in protocol Testing," Proc. PCCS, Seoul, 1985.
10. H. Zimmermann, "On protocol engineering," Proc. IFIP, pp.283-292, Paris, 1983.



최 양 희

저자 약력

- 1955. 7. 27 일생
- 1975 서울대 공과대학전자과 졸업
- 1977. 한국과학기술원 전자과 졸업
- 1984. 프랑스ENST 공학박사
- 1980~1984 프랑스주립전기통신연구소 연구원
- 1977~현재 한국전자통신연구소 망 기술연구 실장