2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

복잡한 통신 프로토콜의 정형검증 방법 연구

이주용, 방기석, 최진영
고려대학교 컴퓨터학과
e-mail : {jlee,kbang,choi}@formal.korea.ac.kr

A Study on Formal Verification of complex network protocol

JooYong Lee, Ki-Seok Bang, Jin Young Choi
Dept. of Computer Science & Engineering, Korea University

요 약

시스템이 과거와 달리 통신을 이용한 분산 환경으로 많이 구현되고 있다. 이에 따라 각 시스템 사이에 통신 메커니즘에 따라 그 안정성 및 효율이 크게 좌우된다. 현재 매우 다양한 통신 메커니즘이 제안되고 사용되고 있으나 점점 더 그에 대한 설계 및 이해가 복잡해지고 있는 실정이다. 따라서 복잡한 통신 메커니즘에 대한 정확성 검증이 매우 중요하다. 본 연구에서는 Spin 을 이용하여 복잡한 통신 프로토콜에 대해 모델링 및 정형 검증에 대해 논한다.

1. 서론

시스템이 과거와 달리 통신을 이용한 분산 환경으로 많이 구현되고 있다. 이에 따라 각 시스템 사이에 통신 메커니즘에 따라 그 안정성 및 효율이 크게 좌우된다. 현재 매우 다양한 통신 메커니즘이 제안되고 사용되고 있으나 점점 더 그에 대한 설계 및 이해가 복잡해지고 있는 실정이다. 따라서 복잡한 통신 메커니즘에 대한 정확성 검증이 매우 중요하다. 그러나 전통적인 테스팅 기법만으로는 발생 가능한 모든 상태에 대한 검사가 불가능하다. 이에 본 연구에서는 분산환경의 통신 프로토콜의 정형 검증[1]을 위해 개발된 Spin[3]을 이용하여 복잡한 통신 메커니즘을 모델링하고 그 정확성을 검증해보았다. 본 논문은 다음과 같이 구성된다. 2 절에서는 정형기법과 Spin 에 대해 간략히 소개하고 이어, 3 절부터는 복잡한 통신 메커니즘의 예로 Abracadabra Protocol 을 설명하고 이를 모델링하여 검증한다.

2. 정형기법과 Spin

정형기법[1]은 수학과 논리학에 기반을 둔 방법으로 하드웨어나 소프트웨어 시스템을 명세하거나 검증하는 것이다. 수학적 기호를 사용하여 시스템의 명세를 하고 검증할 특성 또한 논리로 기술하여 시스템이 특성을 만족하는지를 수학적 성질을 이용하여 검증하므로 자연어가 내포하는 애매모호함이나 불확실성을 최소한 줄일 수 있다. 따라서 복잡한 시스템에 어떤 특성이 만족하는지를 검증하여 최소한으로 검증된 특성에 대해서는 믿을 수 있고 사용할 수 있다.

정형 기법은 정형 명세와 정형 검증으로 나뉜다. 정형 명세는 시스템이 달성해야 할 요구사항과 그러한 요구사항을 완성 할 수 있는 설계를 묘사하는 데 목적이 있다. 그리고 정형 검증은 명세가 정확한지, 즉 설계가 요구사항을 만족하는지를 증명하는데 목적이 있다. 정형 기법은 쉽게 찾아내기 힘든 불일치성, 애매 모호함, 불완전성을 나타내 보임으로써 보다 완벽한 시스템을 구축 할 수 있게 한다.

SPIN[2]은 AT&T Bell 연구소에서 1995 년에 발표한 LTL 모델체커이다. 기존의 모델 체킹 방법을 개선하기 위해 On-the-Fly 방식을 사용하여 메모리의 효율적인 사용을 가능하도록 하였다. SPIN 은 비동기적 프로세스 시스템(asynchronous process system)의 설계와 검증(verify)을 지원하는 가장 일반적인 tool 이다. SPIN 은 또한 분산 소프트웨어(distributed software)와 통신 프로토콜(communication protocol) 검증에 아주 유용하게 사용되고 있다. SPIN 검증 모델은 프로세스 상호작용(Process interactions)의 정확성(correctness)에 초점을 둔다. SPIN 은 분산 시스템에서의 논리적 설계 오류를 찾는데 사용될 수 있다. 특히 운영체제, 데이터 통신 시스템, 교환 시스템(Switching Systems), 동시성 알고리즘(Concurrent algorithms), 철도 신호 프로토콜(railway signaling protocols) 등 실제적인 시스템의 디자인에 매우 적합한 것으로 평가되고 있다. 이 도구를 이용해서 명세(specification)의 일관성(consistency)을 검사할 수 있는데 데드락(Deadlock), 명세되지 않은 반응(Unspecified receptions), flags incompleteness, 경쟁 조건(race conditions)등을 잘 보여준다. 선형시제논리(Linear Temporal Logic:LTL)[4]이나, next-time free LTL, 또는 Buchi Automata 를 이용해서 정확성(correctness properties)을 검증한다.

SPIN 은 PROMELA(Process Meta LAnguage) [3]라는 검증 언어를 사용하여 설계를 하고 PROMELA 는 LTL 의 구문으로 명세화 하여 정확성을 증명한다. PROMELA 는 검증 모델 언어(Verification modeling language)이고, 분산 시스템(distributed system)이나 프로토콜의 추상모델(abstraction)을 제공하고 프로세스(processes), 메시지 채널(message channels),

그리고 변수들로 구성되어 있다. SPIN 에서는 선형시제논리(LTL)을 이용하여 검증을 한다.

3. Abracadabra Protocol

Abracadabra[5]는 K.J. Turner 가 정형 명세 언어를 평가하기 위한 기준으로 제시한 통신 예제의 일종으로 Abracadabra Service 와 Abracadabra Protocol 로 구성되어 있다. Abracadabra Service 는 연결지향형(connection-oriented) 서비스로 통신 사용자간의 연결을 설정하고 신뢰성 있게 데이터를 전송할 수 있게 해 준다. 이를 위해 Abracadabra Service 가 제공해주는 primitive 는 다음과 같다. SDU 는 Service Data Unit 의 약자로 여기에 데이터가 포함되며 DatReq 와 DatInd 에만 인자로 첨가된다.

단계	용도	이름	인자
Connection	Request	ConReq	-
	Indication	ConInd	-
	Response	ConResp	-
	Confirmation	ConConf	-
Data	Request	DatReq	SDU
	Indication	DatInd	SDU
Disconnection	Request	DisReq	
	Indication	DisInd	

표 1 Service Primitives

Abracadabra Protocol 은 앞서 설명한 Abracadabra Service 아래 계층에서 데이터 흐름을 제어하고 에러를 감지하면서 신뢰할 수 없는 전송매체를 매개로 데이터를 주고받을 수 있게 해 주는 역할을 한다. Abracadabra Protocol 의 모형은 아래 그림과 같다.

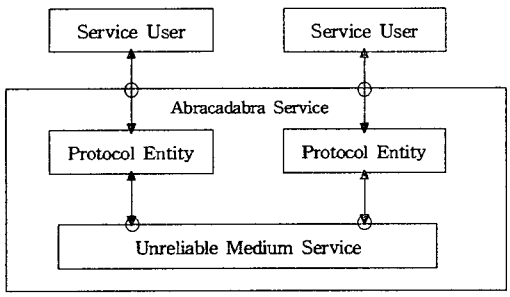


그림 1 Abracadabra Protocol

Abracadabra Protocol 이 이용하는 흐름 제어 방법은 기본적으로 Sliding Window Protocol 에서 이용하는 방법과 같다. 송신자가 PDU(Protocol Data Unit)에 sequence number 를 첨부하며 전송하면 수신자는 sequence number 를 이용하여 원하는 데이터인지 여부를 판단한다. 원하지 않는 데이터로 판별되면 패킷을 버려버리고 그렇지 않다면 다음 자료를 요구한다. 하지만 윈도우 크기를 2 로 제한함으로써 sequence number 가 0 과 1 을 반복하게 된다. 즉, Abracadabra Protocol 은 일종의 Alternating Bit Protocol 이다. 한편 연결 설정 요구, 자료 전송 요구, 연결 해지 요구에 대해 상대측에서 응답이 없으면 재전송을 시도하고 일정회수만큼 재전송을 시도한 후에도 응답이 없으면 연결을 끊어버리고 초기 상태로 돌아간다. 또한 이미 설정된 연결 경로를 통해 데이터를 주고받는 동안 상대편이 연결 설정을 요구하는 부적절한 상황이 발생하면 연결을 끊고 양측 모두를 초기 상태로 돌려놓는다. 이러한 동작은 다음과 같은 protocol data unit 을 이용하여 일어난다.

목적	이름	인자	관련된 Service Primitive
Connection Request	CR	-	ConReq, ConInd
Connection Confirmation	CC	-	ConResp,ConConf
	DT	SDU	DatReq, DatInd
		Sequence Number	
Acknowledgement	AK	Sequence Number	-
Disconnection Request	DR	-	DisReq, DisInd
Disconnection Indication	DC	-	-

표 2 Protocol Data Unit

Abracadabra Protocol 은 연결설정, 자료전송, 연결해지,에러 발생의 상태에 있을 수 있으며 상태의 전이는 현재 상태와 protocol data unit, service primitive 에 따라 달라지게 된다. 아래 그림을 보면 protocol data unit 과 service primitive 에 따른 상태 변화가 어떻게 일어나는지를 알 수 있을 것이다.

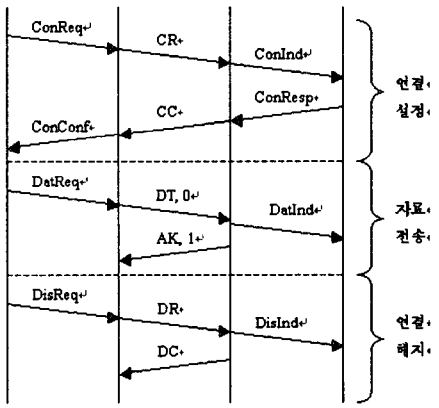


그림 2 Abracadabra 의 상태 변화

정형 검증을 위해 Abracadabra Protocol 의 상태 추이를 좀 더 정형적으로 세분화하여 DFA(Deterministic Finite Automata)로 나타내면 아래 그림과 같다.

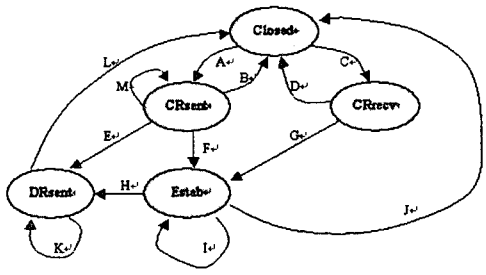


그림 3 Abracadabra Protocol 의 상태전이도

■ Closed : 초기 상태

- CRsent : Closed 상태에서 CR 을 보낸 후의 상태
- CRrecv : Closed 상태에서 CR 을 받은 후의 상태
- DRsent : DR 을 보낸 후의 상태
- Estab : 연결이 설정된 상태
- A: ConReq
- B: DR
- C: CR
- D: DisReq / DR
- E: DisReq / timeout[2]
- F: CC / CR
- G: ConResp
- H: DC / CC / DisReq / AK[2] / CR[2] / timeout[4]
- I: DatReq / AK[1] / DT / CR[1] / timeout[3]
- J: DR
- K: timeout[5]
- L: DC / DR / timeout[6]
- M: timeout[1]
- timeout[1]: CR 이 전송매체에서 손실되고 재전송을 할 수 있다.
- timeout[2] : CR 이 전송매체에서 손실되고 재전송 시도회수가 재전송 허용회수를 초과하였다.
- timeout[3]: DT 가 전송매체에서 손실되고 재전송을 할 수 있다.
- timeout[4] : DT 가 전송매체에서 손실되고 재전송 시도회수가 재전송 허용회수를 초과하였다.
- timeout[5]: DR 이 전송매체에서 손실되고 재전송을 할 수 있다
- timeout[6]: DR 이 전송매체에서 손실되고 재전송 시도회수가 재전송 허용회수를 초과하였다.
- AK[1] :sequence number 가 올바르다.
- AK[2] :sequence number 가 올바르지 않다.
- CR[1]: 연결 설정 후 아무런 사용자 데이터를 주고받지 않았다.
- CR[2]: 연결 설정 후 사용자 데이터를 주고받은 적이 있다.

4. Spin 을 이용한 모델링

앞서 Abracadabra 는 서비스 계층과 프로토콜 계층으로 구분되어진다고 언급하였다. 서비스 계층은 프로토콜 계층보다 상위에 있게 된다. 프로토콜 계층은 전송 매체와 같은 물리 계층을 매개로 하여 통신을 하고, 서비스 계층은 프로토콜 계층을 매개로 통신을 하는 것으로 볼 수 있다. 따라서 Abracadabra 시스템을 검증하려면 물리 계층을 매개로 한 프로토콜 계층간의 통신을 검증하고 그 후에 프로토콜 계층을 매개로 한 서비스 계층간의 통신을 검증하면 된다. 비록 개념상으로는 한번에 모든 계층을 검증하는 것도 가능하지만 상태폭발에 빠질 가능성이 높은 문제가 있다. 본 논문에서는 일단 신뢰할 수 없는 물리 계층을 매개로 한 프로토콜 계층을 검증하였다.

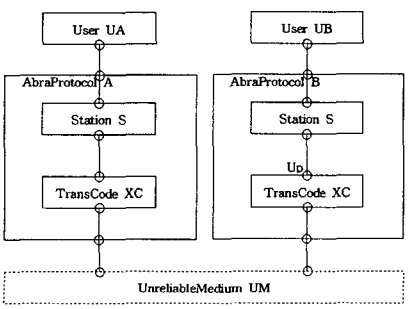


그림 4 Abracadabra Protocol 의 검증 모델

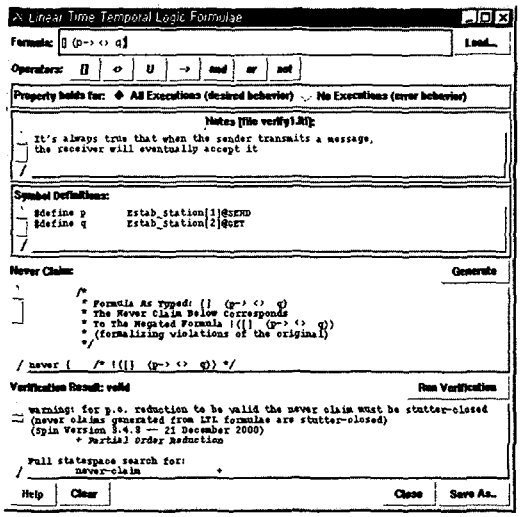
위 그림은 Abracadabra Protocol 의 검증 모델이다. 기본적으로 각 모듈은 프로세스로 구현되며 모듈간의 연결은 채널로 구현된다. Station 모듈은 Abracadabra Protocol 을 나타낸 것으로 가장 핵심적인 모듈이다. 이 모듈은 5 개의 프로세스로 구성된다. Abracadabra Protocol 은 **그림 3** 에서와 같이 5 개의 상태로 구성된 DFA 로 표현되기 때문이다. 한 프로세스에서 5 개의 상태로 전이하도록 하지 않고 상태의 전이가 프로세스의 전이가 되도록 한 것은 Spin 이 partial order reduction 을 이용하는 것을 염두해둔 결정이다. Spin 은 현재 실행하고 있는 프로세스(sleep 상태가 아닌 run 상태의 프로세스)에서 갈 수 있는 경로만을 탐색하므로 일반적으로, 덩치가 큰 하나의 프로세스가 있는 것보다는 조그만한 여러 개의 프로세스가 있는 상황에서 효율적으로 작동한다. Station 모듈 이외의 나머지 모듈들은 단순하기 때문에 하나의 프로세스로 구현하였다. Station 모듈을 구성하는 5 개의 프로세스는 User 모듈과 TransCode 모듈과 통신하기 위해 2 개의 채널을 공유하지만, 현재 상태에 해당하는 프로세스만 채널을 이용할 수 있도록 해야 한다. 즉, 현재 상태가 Estab 이면(**그림** 3 참조) Estab 에 해당하는 프로세스만 채널을 이용할 수 있어야 한다. TransCode 는 물리계층을 나타내는 모듈이다. 이 모듈은 Station 모듈에서 받아온 패킷을 UnreliableMedium 모듈로 보내고 반대로 UnreliableMedium 모듈로부터 패킷을 받아 Station 모듈로 넘겨준다. UnreliableMedium 모듈은 TransCode 모듈로부터 패킷을 받아 반대편 TransCode 모듈로 전달하던가 패킷을 잃어버린다. 즉 신뢰성 없는 전송매체인 것이다. 이 모듈에서 일어나는 패킷 손실은 Station 모듈에서 일어나는 재전송의 원인이 된다. 마지막으로 User 모듈은 Abracadabra Service 를 이용하는 모듈이다. 이 모듈에서 service primitive 를 발생시키고 처리한다. 본 논문에서는 서비스가 아닌 프로토콜에 대한 검증을 다루므로 User 모듈에 해당하는 프로세스는 필요가 없다. 하지만 테스트를 위해 간단한 User 프로세스를 만들어보았다. 이에 대한 사항은 다음 절에서 다루겠다. 구체적인 Promela 코드는 지면관계상 생략한다.

5. Spin 을 이용한 검증

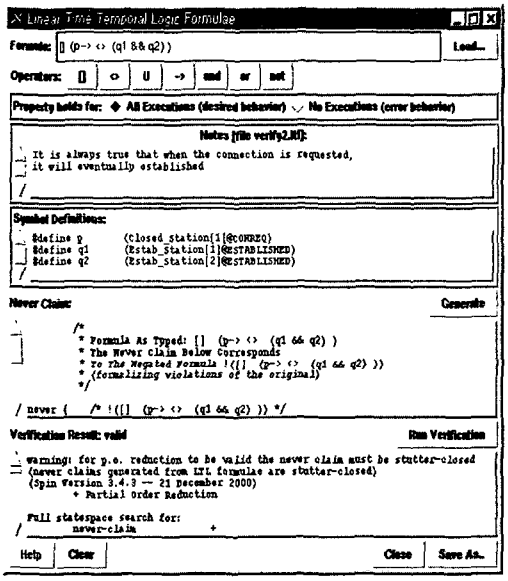
프로토콜을 검증할 때 시도해볼 수 있는 것 중의 하나는 가상의 사용자 프로세스를 만드는 것이다. 본 논문에서는 한 쪽 사용자는 전송만 하고 다른 한 쪽 사용자는 수신만 하는 경우에 대해 테스트를 해 보았다. Spin 을 이용해 Safety 검사를 수행해보아 이 경우 에러가 없음을 알 수 있었다. 아래는 검증 결과의 일분이다. 에러가 없음을 볼 수 있다.

```
Full statespace search for:
        never-claim             - (not selected)
        assertion violations    - (disabled by -A flag)
        cycle checks            - (disabled by -DSAFETY)
        invalid endstates       +

State-vector 340 byte, depth reached 1882, errors: 0
   39937 states, stored
   54120 states, matched
   94057 transitions (= stored+matched)
      14 atomic steps
hash conflicts: 1579 (resolved)
(max size 2^19 states)

Stats on memory usage (in Megabytes):
13.898  equivalent memory usage for states (stored*(State-vector +
overhead))
10.650  actual memory usage for states (compression: 76.63%)
        State-vector as stored = 259 byte + 8 byte overhead
2.097   memory used for hash-table (-w19)
0.240   memory used for DFS stack (-m10000)
13.090  total actual memory usage
```

통신 프로토콜은 송신자가 보낸 데이터가 반드시 언젠가는 수신자에게 전달된다는 것을 보장해주어야 한다. 아래 그림은 Abracadabra Protocol 이 이러한 특성을 만족함을 보여준다.



ConReq 가 들어오면 양쪽 State 모듈이 Estab 상태로 가야한다는 성질도 아래와 같이 만족시킨다.



6. 결론

현재 매우 다양한 통신 메커니즘이 제안되고 사용되고 있으나 점점 더 그에 대한 설계 및 이해가 복잡해지고 있는 실정이다. 따라서 복잡한 통신 메커니즘에 대한 정확성 검증이 매우 중요하다. 그러나 전통적인 테스팅 기법만으로는 발생 가능한 모든 상태에 대한 검사가 불가능하다.

본 연구에서는 정형기법 도구인 Spin 을 이용하여 통신 프로토콜을 모델링하고 검증하였다. 자연어나 그림으로는 이해시키기 어려운 복잡한 통신 메커니즘을 명확하고 쉽게 이해할 수 있었다. 또한 자연어나 그림으로 나타나지 않는 에러를 Spin 으로 모델링하면서 찾아낼 수 있었다. 아울러 손쉽게 통신 프로토콜의 정확성 여부를 확인해볼 수 있었다.

참고문헌

[1] Edmund M. Clarke and Jeannette M. Wing, "Formal Method : State of the Art and Future Directions", ACM Computing Surveys, pp. 626-643

[2] Gerald J. Holzmann, "The Model Checker SPIN", IEEE Transactions on Software Engineering, VOL. 23, NO 5, MAY 1997, pp279 – 295.

[3] Gerald J. Holzmann, "Design and Validation of Computer Protocols", Prentice Hall, 1991.

[4] Zohar Manna, Amir Pnueli, "The Temporal Logic of Reactive and Concurrent Systems – Specification", Springer-Verlag, 1996

[5] Kenneth J.Turner, "Using Formal Description Techniques",John Wiley & Sons, 1993.