OSPF 프로토콜 오류 검출기 개발

홍창표° 김태형 이현성 차원수 정세훈
금오공과대학교 컴퓨터 공학과
{cphong°, thkim, hslee, wscha, shjung}@cespc1.kumoh.ac.kr

The Development of an OSPF Protocol Error Detector

Changpyo Hong° Tae-Hyong Kim, Hyun-Sung Lee, Won-Soo Cha, Se-Hoon Jung
Kumoh National Institute of Technology

요 약

수동 시험은 기존의 프로토콜 검증 방법과는 달리 제품이 실제 운용되는 환경에서 입출력의 관찰만을 가지고 제품의 오류를 검출하는 시험이다. 본 논문은 EFSM(Extended Finite State Machine) 모델을 이용하여 homing 기법을 사용하는 수동시험 기법을 제안하고 이를 검증하기 위하여 제안된 수동 시험 기법을 사용하는 OSPF 프로토콜의 오류 검출기를 개발하였다. 개발된 오류 검출기는 Zebra[6]와 시스코 라우터로 구성된 소규모 네트워크에서 Zebra를 이용해 발생시킨 오류 OSPF 패킷을 잘 검출해내었다.

1. 서 론

통신망이 비대해 짐에 따라 네트워크는 다양하고 복잡한 구조를 가지게 되었고 통신 프로토콜도 무수히 많이 생성되었다. 이러한 통신 프로토콜 제품이 실제 네트워크 환경에서 동작할 때, 기존의 통신 제품과 아무런 문제없이 잘 동작하기 위해서는 설계/구현 과정에서 표준 규격에 따라 정확하게 제작되었는지에 대한 검증의 중요성이 커지고 있다. 이에 여러 가지 검증 방법이 표준화 되었고[1] 프로토콜의 효과적인 설계와 검증/시험/구현을 연구하는 프로토콜 공학 분야가 80년대 후반부터 시작하여 90년대를 거쳐 좋은 연구 성과를 보여주고 있다. 하지만 프로토콜 공학이 주로 이론적인 분야에 치우쳐져 있고 실제 연구결과를 프로토콜 개발과정에 적용하는 실제적인 연구 분야에서는 몇몇 실험적인 결과들 외에는 큰 성과를 거두지 못하고 있다.

최근 이러한 문제점을 해결하기 위한 방법으로 수동시험이 연구되고 있다. 구현제품의 검증을 출시하기 전에 구현제품에 입력되는 각종 입력에 대한 제어를 수행하여 처리하는 검증방법은 추가적인 작업과 비용을 필요로 한다는 단점이 있다. 그러나 수동시험은 구현된 프로토콜을 실제 네트워크에 적용하여 운영한 후에 발생하는 오류를 체크하는 방법이므로 이러한 단점을 극복할 수 있는 기법이다.

수동시험 기법은 대상 프로토콜을 FSM(Finite State Machine) 또는 EFSM(Extended FSM)으로 모델링 한 프로토콜 에뮬레이터를 구현하고 실제 운용중인 네트워크 장비의 트래픽을 모니터링 한다. 이때 대상 프로토콜 패킷의 을 캡처해서 내용을 분석하여 오류 여부를 판별한다. 본 논문에서는 OSPF(Open Shortest Path First)[2] 프로토콜에 수동시험 기술을 적용한 OSPF 프로토콜 오류 검출기 개발에 대해 기술한다.

2. 수동시험

2.1 수동시험의 개념

지금까지 연구된 수동시험 기법은 다음과 같이 몇가지 종류로 나누어 볼 수 있다.[3] Homing 방법은 두 단계의 과정을 통해 오류를 검출한다. 먼저 시험대상 프로토콜이 현재 어느 상태에 있는지를 확인하는 수동 유도열을 얻어내는 과정이다. 이후, 입출력 정보를 통해 프로토콜 모델을 분석하여 구동 가능한 상태를 좁혀나감으로써 최종 상태 하나에 이르렀을 때 그 상태에서 프로토콜의 모델을 확인하여 오류를 검출하는 것이다.[4][5] Invariant 기법은 EFSM 명세로부터 일련의 성질을 추출하고 구현 제품으로부터 얻어낸 트레이스가 이러한 일련의 성질에 부합되는지를 확인하여 오류를 검출하는 것이다[6]. Backward tracking 기법은 먼저 트레이스를 명세에 의거 하여 트레이스를 역순으로 추적한 다음 다시 트레이스를 과거로 역추적하여 시스템에 부합되지 않는 오류를 검출한다[7].

2.2 수동시험 알고리즘

본 논문에서는 EFSM을 대상으로 하여 Homing 기법을 사용하는 수동시험 기법을 제안하여 사용 한다. 먼저 대상 EFSM을 다음과 같이 정의한다.

[정의 1] EFSM $M = (S, V, P, I, O, T)$이고 S, I, O, T는 각각 유한하며 공집합이 아닌 논리 상태, 입력 신호, 출력 신호, 천이의 집합이고 V, P는 각각 유한한 내부 변수 및 입력 매개변수의 집합이다. 천이의 라벨 $t(\in T)$는 다시 다음과 같이 (s_s, g_I, g_D, op, s_f)로 정의된다. s_s는 t의 시작상태, g_I는 입력신호에 대한 가드로 (i, P^i, g_{P_i})로 나타내어진다. 여기서 $i \in I$, $P^i \subset P$, g_{P_i}는 입력 매개변수의 가드로 널이거나 $V'(\subseteq V)$와 $P'(\subseteq P)$의 원소로 나타내어지는 논리식으로 표현된다. g_D는 도메인 가드로 널이거나 $V''(\subseteq V)$의 원소로 나타내어지는 논리식으로 표현된다. op는 순차적인 동작으로 출력문이나 할당문과 같은 간단한 문장으로 구성되며, s_f는 t의 종료상태이다.

EFSM 기반 수동시험에서는 내부 변수 처리 부분의 오류를 찾아내기 위해서 구현 제품의 상태뿐만 아니라 내부 변수의 값도 계속 확인을 해야 한다. 이를 형식적으로 용이하게 표현하기 위해서 다음과 같은 용어를 사용한다. 먼저 EFSM에서 천이의 도메인 가드에 사용된 변수를 제어변수라 할 때, Δ는 V내의 모든 제어변수에 의해 만들어지는 도메인을 나타내고, $\wedge$는 P내의 모든 입력 매개변수에 의해 만들어지는 도메인을 나타낸다. 이때 한 상태 s에서 허용되는 Δ의 부분집합을 그 상태의 도메인이라 하고 $d(s)$라 나타내자. DOM 연산자는 논리식을 취해서 그 조건을 만족시키는 Δ의 부분집합을 도출해주며 COND 연산자는 Δ의 부분집합을 취해서 해당하는 논리식을 구성해준다고 한다. 한 천이 t_i의 선조건은 P_i로 나타내며 t_i의 도메인 가드 g_D를 의미하고, t_i의 매개변수조건은 λ_i로 나타내며 t_i의 입력 매개변수 가드 g_p를 의미한다. t_i의 후조건은 $Q_i: \Delta \times \wedge \rightarrow \Delta$로 나타내며 t_i의 동작 op_i에 따라 t_i의 시작상태에서의 도메인 변화를 도출하는 함수이다.

[알고리즘 1] 수동시험에서 EFSM의 오류 검출을 위한 알고리즘
입력 : (1) 명세 EFSM A, (2) 구현제품의 EFSM B로부터
관찰된 I/O열 : $i_1/o_1,...,i_k/o_k,...$
출력 : 구현제품 B의 구현 오류 여부

```
begin
  L1 = {s1,...,sn};   /* 현재 상태 후보 */
  d(L) = Δ;  /* 현재 상태 도메인 후보 */
  while GetNextInput(i,val(P^i)) do
    L2 = φ;                    /*val(P^i)는P^i의 값 관찰*/
    for each transition t_i such that i(t_i)=iΛs_ST(t_i)∈L1 do
     if d(L) ⊆ P_i ∨ (d(L)⊄P_i ∧ d(L)∩P_i ≠ φ)
          L2 = L2 ∪ s_FN(t_i);
          d(s_FN(t_i)) = Q_i(d(L),val(P^i));
     end
     d(L) = ∪_i d(s_FN(t_i)); according to R1 and R2;
     if L2 = φ then return FAULT;
     else L1 = L2;
  end
end
```

3. OSPF 오류 검출기 개발

제안된 EFSM 대상 성능을 검증하기 위하여 본 논문은 현재 네트워크 자원 관리용으로 널리 사용되고 있는 라우팅 프로토콜인 OSPF 프로토콜을 검증 대상으로 하는 오류 검출기를 개발하였다.

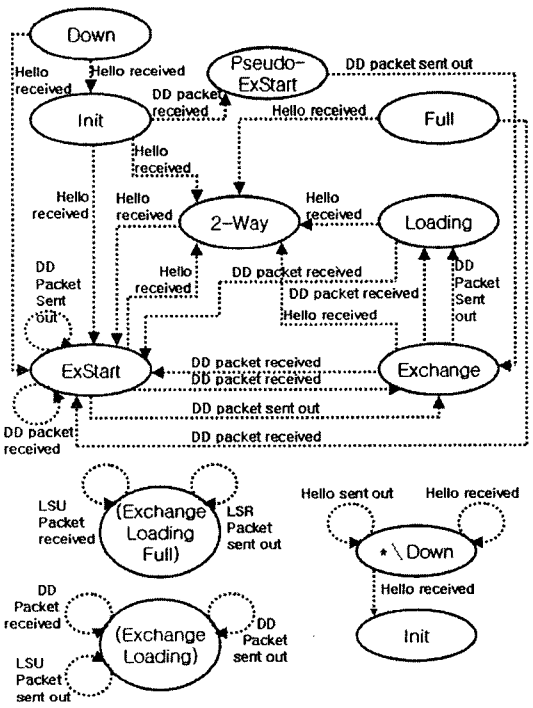

그림 1 : OSPF 프로토콜의 상태 천이 다이어그램

3.1 OSPF 모델링

OSPF의 상태 천이 다이어그램은 그림1과 같다. Exstart는 인접 라우터 간에 LSA 절차를 최초로 시작할 때 사용된다. Init는 Hello 패킷이 수신될 경우 사용된다. Down은 인접 라우터와 통신이 있을 경우 사용된다. 2-Way는 인접 라우터 간에 양방향 통신이 될 경우 사용된다. Exchange는 인접 라우터 간에 라우팅 테이블을 교환하여 서로 비교할 때 사용된다. Loading은 Exchange 상태에서 좀 더 최근의 LSA를 요구하려고 할때 사용된다. Full은 LSA 절차가 모두 완료 되었을 때 사용된다. 나머지 Pseudo-ExStart , Exchange-Loading -Full과 , *\Down , Exchange Loading Init 상태는 LSA 절차에 필요한 패킷인 Data Description , Link State Request , Link State Update 패킷을 처리하기 위해 사용된다.

3.2 오류 검출 환경

오류검출을 위한 모니터링 환경은 그림2와 같다. 기본적인 LSA 절차를 수행하기 위해서는 라우터 두 대가 필요하다. 이 실험에서는 수동유도 과정을 처리하지 않고 이미 처리되어 오류가 발생하는 지점을 찾았다는 가정에서 오류를 검출 하였다. 그림1에서 발생시킨 이벤트를 처리할 때 명세서에서 허용할 수 없는 두 가지 형태의 오류를 발생시켰다. 첫째, 특정 상태에 진입했을 때 그 상태에서 다음 상태로 전이할 경우에 입력인자가 잘못된 경우를 구별해 내는 것이다. 예를 들면 명세서는 Init 상태에서 ExStart 상태로 전이 될 때 입력될 이벤트는 Hello 패킷이 수신되었다는 이벤트이지만 Hello 패킷을 다른 라우터로 송신하는 이벤트를 강제로 입력하여 오류를 발생시킨 경우이다. 둘째, 제대로 된 입력과 상태에서 명세서에 위반되는 다른 상태로 전의 되는 오류를 발생시켰다. 예를 들면 명세서에는 Full 상태에서 Hello 패킷 수신 이벤트가 입력될 경우 2-Way 상태로 전이되어야 한다고 명시되어 있다. 하지만 이 경우에 Full 상태에서 절대로 전이될 수 없는 Loading 상태로 전이되도록 오류를 발생시킨 경우이다. 이 두 가지 오류는 패킷의 헤더 필드에서 패킷 자신의 종류를 나타내는 필드를 수정하여 처리하였다.

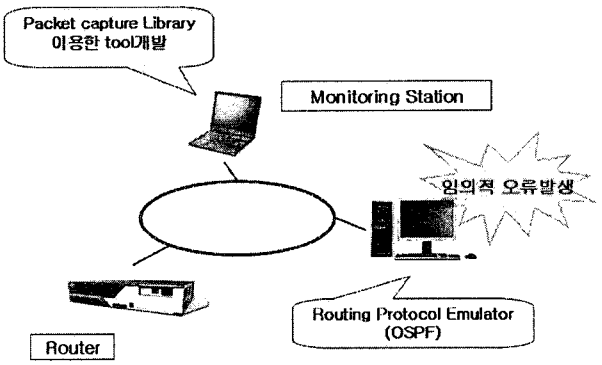

그림 2 : 실험용 네트워크 구조도

그러나 오류를 발생시키기 위해서는 OSPF 패킷을 조작할 수 있어야 한다. 이를 위해서 우리는 PC에 리눅스를 설치하여 라우터 기능을 구현한 Zebra[8] 프로그램을 사용하였다. Zebra는 IOS 기능을 내장하고 있어서 각종 프로토콜을 제어하고, 보안, 경로최적화, 네트워크 관리를 할 수 있다. 그리고 이러한 오류를 검출하기 위해서 텍스트 모드로 패킷을 캡처하는 Tethereal[9] 프로그램을 사용하였다.

3.3 오류 검출기

이 논문에서 제안하는 오류 검출기는 그림3과 같은 구조를 가진다.

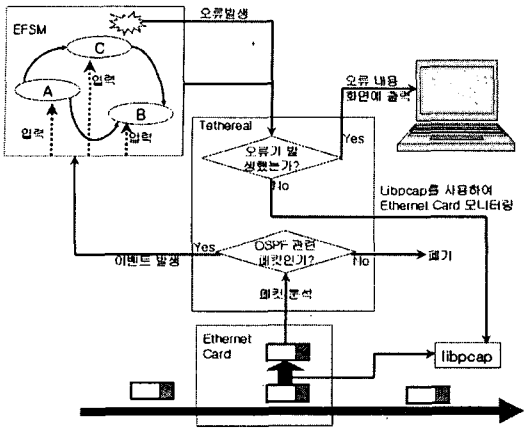

그림 3 : 오류 검출기 구조

LSA 절차에 따라 전송되던 패킷은 오류 검출기의 이더넷 카드에서 libpcap 라이브러리[10]를 통해 내부 버퍼에 복사되고 복사된 패킷은 Tethereal의 여러 가지 필터를 통해 OSPF 관련 패킷만은 추출한다. 추출된 패킷은 Tethereal에서 분석되어 LSA에 관계된 각종 이벤트를 발생시킨다. 이 때 발생된 이벤트와 패킷의 정보를 이용하여 기존에 입력된 EFSM 모델에 대하여 homing 과정을 수행하게 되고 일단 프로토콜의 상태가 확인되면 이후부터 관찰되는 패킷에 대해 EFSM 모델과 비교하여 오류가 발견되면 이를 화면에 표시하게 된다.

4. 평가

구성된 실험 네트워크에서 오류 검출기를 작동시키면 그림4와 같은 정보가 출력된다. 출력화면에서 오류가 검출된 패킷이 출력될 때 상단에 해당 패킷이 가지고 있는 정보(송신자, 수신자, 패킷 타입, DR, BDR....등)를 확인할 수 있다. 하단에 출력되는 내용은 패킷이 처리되기 위해 상태천이를 거치는 동안 오작동이 일어났을 때 천이 과정이 시작된 상태(Strat State)와 천이된 상태(Next State), 발생한 이벤트(Event), 입력 이벤트 매개변수를 이용한 Predicate와 동작(Action)을 확인할 수 있다.

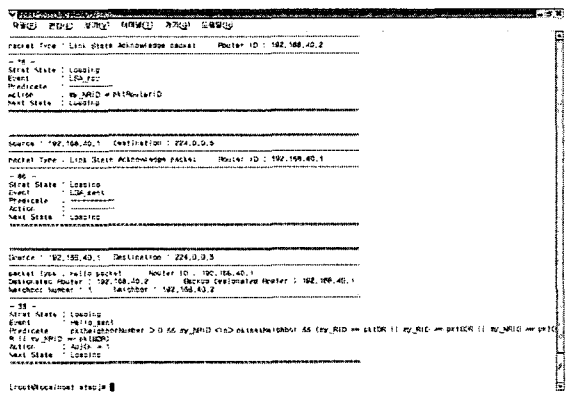

그림 4 : 오류 검출기 출력 화면

Zebra에서 발생시키는 오류는 패킷의 종류가 달라지는 경우와 패킷의 매개변수 값이 달라지는 두 가지 경우가 있다. 개발된 오류 검출기는 이 두 가지 오류에 대해서 대부분의 오류를 잘 검출해 주었다. 한편 개발된 오류 검출기가 오류를 발생시키지 않은 정상적인 경우에도 오류로 판정하는 경우가 있었는데 이는 패킷의 전송이 매우 빠르게 이루어질 경우 패킷 캡쳐 소프트웨어 성능의 한계로 이 중 일부를 잡아내지 못하기 때문에 발생하였다. 따라서 오류 검출기의 신뢰성 향상을 위해서는 고속 패킷 캡쳐를 위해 처리 소프트웨어의 개선이나 하드웨어화가 필요하다고 판단된다.

5. 결론

이 실험에서 우리는 OSPF 프로토콜에 대해 수동시험 알고리즘을 적용하였다. 이때 발생한 오류에 대해 OSPF의 인터페이스 상태 머신을 거치면서 처리되는 과정을 수동시험으로 분석하여 오류검출이 성공적으로 이루어 졌다. 따라서 이 오류 검출기는 다른 프로토콜에서도 정상적으로 작동할 것이라고 생각된다.

또한, 현재 국내에서 통신 프로토콜의 수동시험 분야에 대한 연구가 아직 본격적으로 수행되지 않고 있는 상황인데, 본 연구 과제가 국내에서 관련 후속 연구가 이루어지는 개기가 되어 국내 프로토콜 공학 연구의 활성화에 기여할 것이라고 생각된다.

향후 본 오류 검출기에 사용된 수동시험 기법 알고리즘을 개선하여 오류 검출시간을 단축시키는 연구를 수행할 것이며 오류의 검출만이 아니라 오류가 프로토콜 내의 어느 위치에 존재하는가를 찾아내는 오류 Location 알고리즘을 제안하고 이를 검출기에 적용할 것이다.

6. 참고 문헌

[1] ISO, "OSI Conformance Testing Methodology and Framework", IS-9646, 1991

[2] J. Moy, Ascend Communications, Inc. "OSPF Version 2", RFC 2328, April 1998

[3] David Lee, Dongluo Chen, Ruibing Hao, Raymond E. Miller, Jianping Wu and Xia Yin,"A Formal Approach for Passive Testing of Protocol Data Portions", Proceedings of the 10th IEEE Internal Conference on Network Protocols(ICNP'02), 2002

[4] D. Lee, A. Netravali, K. Sabnani, B. Sugla, A. John, "Passive Testing and Applications to Network Management", Proceedings of the 1997 International Conference on Network Protocols(ICNP'97), 1997

[5] M. Tabourier, A. Cavalli, "Passive Testing and Application to the GSM-MAP Protocols", Information and Software Technology, Vol.41, 1999, pp.813-821

[6] A.Cavalli, C.Gervy, S. Prokopenko, "New Approaches for Passive Testing Using an Extended Finite State Machine Specification", Information and Software Technology, Vol.45, 2003, pp.837-852

[7] B. Alcalde, A. Cavalli, D. Chen, D. Khnu, D. Lee, "Network Protocol System Passive Testing for Fault Management: A Backward Checking Approach", FORTE 2004, Lecture Note in Computer Sciences, Vol.3235, pp.150-166, 2004

[8] "http://www.zebra.org/", IP Infusion

[9] "http://www.ethereal.com", Gerald Combs

[10] 노광민, "Libpcap 사용하기", 2002.01.01, KLDP