# 실험계획법을 이용한 TCP 데이타 부분에 대한 상호운용성 시험스위트 생성

## (Interoperability Test Suite Generation for the TCP Data Part using Experimental Design Techniques)

유 지 원 † 김 명 철 †† 설 순 욱 † 강 성 원 ††† 이 영 희 †††† 이 근 구 †††††
(Jiwon Ryu)(Myungchul Kim)(Soonuk Seol)(Sungwon Kang)(Younghee Lee)(Keunku Lee)

**요 약** 통신 프로토콜의 상호운용성을 시험하기 위한 시험경우를 생성하는 방법론이 논문 [1, 2, 3]에서 제안되었고 TCP와 ATM 프로토콜에 적용되었다. 이들 방법론으로 생성된 시험경우는 제어 부분만을 고려하고 있다. 그러나 실제 시험에 있어서는 데이타를 고려하여야 하며, 완전한 시험이 되기 위해서는 이들 변수들이 가질 수 있는 모든 값에 대한 시험을 거쳐야 하지만 이것은 현실적으로 불가능하다. 본 논문에서는 데이타를 포함하는 시험을 하기 위하여 논문 [1]에서 도출한 시험경우에 제어 부분과 데이타 부분을 동시에 고려하여 TCP의 연결 설정 절차에 대한 시험경우를 도출한다. 이 과정에서 시험경우의 수가 너무 커지는 것을 피하기 위해 최소의 실험횟수로 최대의 정보를 얻을 수 있도록 실험계획법(experimental design)을 이용한다. 실험계획법은 적합성 시험에는 적용되었지만 아직 상호운용성 시험에는 적용되지 않았다. 이 방법을 통해 데이타 부분을 포함하는 상호운용성 시험경우를 생성하고, 시험의 검출력 (power of test)을 유지하면서 최소화된 시험경우로 상호운용성 시험을 할 수 있는 가능성을 보인다.

***Abstract*** Test derivation methods suitable for interoperability testing of communication protocols were proposed in [1, 2, 3] and applied to the TCP and the ATM protocols. The test cases that were generated by them deal with only the control part of the protocols. However, in real protocol testing, the test cases must manage the data part as well. For complete testing, in principle we must test all possible values of data part although it is impractical to do so. In this paper, we present a method generating the interoperability test suite for both the data part and the control part of protocols with the example of TCP connection establishment. In this process, we make use of experimental design techniques from industrial engineering to minimize the size of test suite while keeping testing capability. Experimental design techniques have been used for protocol conformance testing but not for interoperability testing so far. We generate the test suite for data part by this method and show a possibility that we can test interoperability of protocols with the minimum number of test cases while maintaining the testing power.

† 학생회원 : 한국정보통신대학원대학교 공학부
jwryu@icu.ac.kr
suseol@icu.ac.kr
†† 종신회원 : 한국정보통신대학원대학교 공학부 교수
mckim@icu.ac.kr
††† 종신회원 : 한국통신 연구개발본부 연구원
kangsw1@kt.co.kr
†††† 정 회 원 : 한국정보통신대학원대학교 공학부 교수
yhlee@icu.ac.kr
††††† 종신회원 : 한국전자통신연구원 네트워크장비시험센터 연구원
kklee@netc.etri.re.kr
논문접수 : 2000년 3월 6일
심사완료 : 2000년 8월 7일

## 1. 서 론

정보통신 관련 장비 및 서비스의 급격한 증가와 더불어, 이들 간의 상호운용성을 보장하는 것은 중요한 문제가 되고 있다. 이를 보장하기 위하여 적합성 시험이 ITU-T [4]와 ISO [5]에서 표준화되어 사용되고 있다. 프로토콜 적합성 시험은 표준에 따라 구현된 하나의 피시험 대상 (Implementation Under Test : IUT)이 해당 표준 및 규정된 요구조건에 적합한지 여부를 확인하는 기술적인 행위로서, 프로토콜 구현물 간의 상호운용성에 대한 가능성을 높여준다. 그러나 규격 내부의 필수

사항, 선택사항, 파라미터 값의 조정 범위 등의 다양성으로 인해 적합성 시험에 합격한 장비 및 서비스들 간에도 실제 통신 환경 하에서 상호운용시에는 제대로 동작하지 않는 현상이 종종 발생하게 된다. 따라서 이러한 문제의 보완을 위하여 구현된 두 개 이상의 피시험 대상을 실제 운용 상태하에서 시험하여 해당 기능들의 올바른 동작 여부를 확인하는 상호운용성 시험이 필요하다. 그러나, 현재까지 상호운용성 시험에 관한 국제적인 표준이나 시험체계 및 방법개발의 진척이 미비한 실정이다.

논문 [2]는 통신 프로토콜의 상호운용성 시험경우를 도출하기 위한 알고리즘을 제안하였고, 논문 [1]과 [3]은 TCP 인터넷 프로토콜과 ATM/B-ISDN 신호 프로토콜에 각각 적용하여 시험경우를 도출하였다. 이들 시험경우는 Finite State Machine (FSM)을 바탕으로 프로토콜의 제어 부분만을 고려하였다. 현재까지 프로토콜의 데이타 부분을 고려하여 상호운용성 시험경우를 도출하는 방법에 대해서는 연구되지 않았다.

그러나 대부분의 프로토콜은 다양한 변수가 존재하고 각 변수는 수많은 값을 가질 수 있기 때문에 실제 프로토콜 시험에서는 데이타 부분을 포함하는 시험경우를 도출해야 한다. 데이타 부분에 대한 시험경우를 구해보면 논문 [1]에서 도출한 시험경우 수보다 훨씬 많아지게 된다. 시간과 비용면에서 시험의 효율성을 높이기 위하여 시험경우를 최소화하는 방법으로 최소의 실험횟수로 최대의 정보를 얻을 수 있도록 하는 실험계획법을 이용한다. 실험계획법은 시험에 대한 검출력을 유지하면서 시험경우의 수를 최소화시켜 상호운용되지 않는 프로토콜의 문제점을 발견할 수 있게 한다. 실험계획법은 적합성 시험 [6]에는 적용되었지만 아직 상호운용성 시험에는 적용되지 않았다. 본 논문은 데이타 부분에 대한 시험을 하기 위하여 논문 [1]에서 도출한 제어 부분으로만 구성된 시험경우에 본 논문에서 세운 가정에 따라 데이타 부분을 고려하여 TCP의 연결 설정 절차에 대한 새로운 시험경우를 도출한다. 이 과정에서 실험계획법을 이용하면 그렇지 않은 경우보다 시험경우의 수가 얼마나 감소하는지를 확인한다.

본 논문의 구조는 다음과 같다. 2장에서는 관련 연구로 기존의 TCP 상호운용성 시험경우 도출 방법과 실험계획법, TCP 연결 설정에 대한 명세 사항에 대해 기술한다. 3장에서는 본 논문에서 세운 가정과 실험계획법을 이용하여 TCP 연결 설정 부분의 데이타 부분에 대한 시험경우를 도출하고 시험경우의 수를 최소화시킨다. 마지막 4장에서는 결론과 향후 연구할 내용을 제시한다.

## 2. 관련 연구

이 절에서는 본 논문과 관련된 연구로서, 제어 부분만을 고려하여 도출한 TCP의 상호운용성 시험경우에 대하여 기술하고, 데이타 부분에 대한 TCP 시험경우의 수를 줄이기 위해서 사용하는 실험계획법에 대해서 살펴본다. 그리고 데이타 부분에 대한 시험경우를 도출하고자 하는 TCP 연결 설정 절차에 대한 명세 사항을 기술한다.

### 2.1 제어부분에 대한 TCP의 상호운용성 시험 경우 생성

논문 [1]은 통신 프로토콜의 상호운용성 시험을 위한 시험경우를 생성하는 알고리즘을 프로그램으로 구현하여 TCP에 적용하였다. 구현한 프로그램은 TCP의 FSM을 입력으로 하여 시험경우를 구한다. 그림 1은 논문 [1]에서 표현한 FSM 중에서 연결 설정 절차만을 표현한 FSM이다. 입력은 상위 TCP 응용프로그램에서 주어지게 되어 있으며, 출력은 TCP 패킷 내에 있는 6 비트의 제어 필드인 URG, ACK, PSH, RST, SYN, FIN의 각 비트 중에서 하나 이상이 선택 표시된다. TCP는 선택된 비트들을 통해 연결 설정을 제어하게 된다.

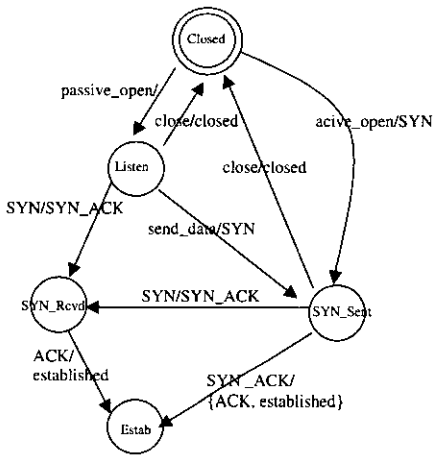


그림 1 연결 설정 절차를 나타내는 TCP FSM

그림 1의 TCP FSM을 입력으로 하여 구현한 프로그램을 수행한 결과, 12개의 상호운용성 시험경우가 도출된다. 이러한 12개의 시험경우는 부록 1에서 보여진다. 다음은 TCP가 연결 설정을 위해 세방향 핸드쉐이킹 (three-way handshaking)을 하는 시험경우로 부록 1의 <1> 시험경우이다.

(Closed,Listen) — active_open_a/[ <,i_SYN,,i_SYN_ACK>,

<established,i_ACK, established,> ] → (Estab,Estab)

(Closed,Listen)는 시작 안정 상태를 의미하고 '/' 앞에 있는 active_open_a는 시험기 A로부터 IUT A로 입력된 입력 심볼을 의미하며, 뒤에 있는 [<,i_SYN,, i_SYN_ACK>, <established,i_ACK,established,>]는 주어진 입력에 대한 출력 심볼 열의 집합이다. 그림 2에서와 같이 각 원소는 <u1,u2,u3,u4>의 백터 형태로 표현될 수 있다. 여기서 u1은 IUT A로부터 인터페이스 A로의 출력, u2는 IUT A로부터 인터페이스 C로의 출력, u3는 IUT B로부터 인터페이스 B로의 출력, u4는 IUT B로부터 인터페이스C로의 출력이다. '→'는 천이(transition)를 의미하고, 뒤이은 (Estab,Estab)는 도착 안정 상태이다. 이러한 시험경우는 데이타 부분의 값이 없기 때문에 실제 시험에 제대로 적용할 수 없다.

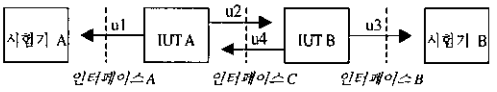


그림 2 두 IUT의 출력 심볼

논문 [3]은 그림 3에서와 같이 상호운용성 시험을 위한 3가지 구조를 제시하였다. 시험 구조 I은 시험기 A가 IUT A로 메시지를 보내면 IUT A가 메시지를 IUT B로 그리고/혹은 시험기A로 보낼 수 있다. 동시에 IUT B는 IUT A로 그리고/혹은 시험기B로 메시지를 보내게

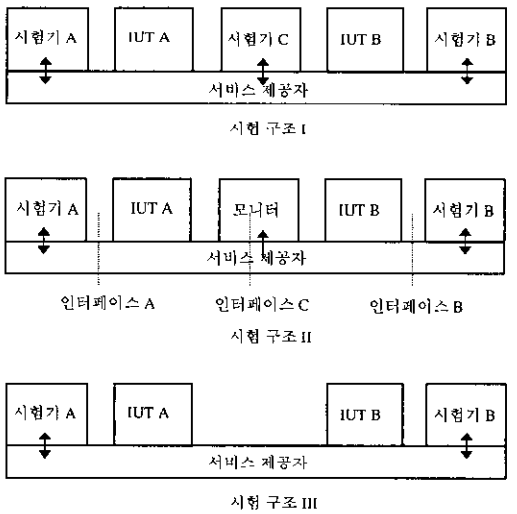


그림 3 상호운용성 시험을 위한 시험 구조

된다. IUT A와 IUT B사이에 존재하는 시험기 C는 IUT A와 IUT B사이에 오가는 내부 신호를 관찰/변경할 수 있다. 시험 구조 II는 IUT A와 IUT B 사이에 오가는 내부 신호를 단지 관찰만 할 수 있다. 반면 구조 III은 시험기나 모니터가 없기 때문에 내부의 오가는 신호를 볼 수 없다. 논문 [1]에서는 제어 부분에 대한 시험경우를 생성하기 위하여 구조 III을 다루었다. 그러나 데이타 부분이 시험경우에 포함되면 IUT A와 IUT B 사이를 오가는 내부 메시지의 변수 값이 어떻게 변화하는지 확인해야 한다. 따라서 본 논문은 구조 II를 채택한다.

### 2.2 실험계획법

실험계획법이란 해결하고자 하는 문제에 대하여 실험을 어떻게 행하고, 데이타를 어떻게 취하며, 어떠한 통계적 방법으로 데이타를 분석하면 최소의 실험횟수로 최대의 정보를 얻을 수 있는가를 계획하는 것이다 [7]

실험을 통하여 얻고자 하는 목적이 명확히 설정되면 인자 (factor)와 인자의 수준 (level)을 선택해야 한다. 실험에 직접 취급되는 원인을 인자라고 부르고 실험을 하기 위한 인자의 조건을 인자의 수준이라고 한다. 인자는 변수라고 볼 수 있고 인자의 수준은 변수에 대한 값으로 볼 수 있다. 예를 들면 온도를 인자로 택한 경우 150° C라든가 200° C라고 하는 값이 그 수준이다. 이렇게 수준으로 취한 값의 수를 인자의 수준수라고 한다.

실험횟수가 과다하게 많은 경우에는 본래의 실험목적을 달성할 수 있는 범위에서 가능한 한 실험의 크기를 줄일 필요가 있다. 고차 (3인자 이상)의 교호작용(interaction)[1]은 일반적으로 유의 (significance)하지 않은 경우가 많으며, 유의하다고 하여도 물리적 해석이 어렵거나 실질적인 행동을 취하기 곤란하다. 따라서 1인자와 2인자 교호작용만을 시험할 뿐이라면 반드시 인자의 모든 조합에서 실험할 필요가 없으며, 불필요한 교호작용이라든가 고차의 교호작용을 구하지 않고 실험의 크기를 작게 할 수 있도록 인자의 조합 중 일부만을 실험하는 방법을 일부실시법 (fractional factorial design)이라고 한다 [7]. 이 실험계획법은 실험에서 취급하고 싶은 인자가 많으나 실험횟수를 가급적 적게 하고자 할 경우에 편리한 방법이다. 그러나 고차의 교호작용은 거의 존재하지 않는다는 가정이 만족되어야 한다. *논문 [8, 9]에 의하면 인자에 의해서 발생하는 대부분의 에러는 1인자나 2인자 교호작용으로 인하여 일어난다. 프로토콜 분야에서도 대부분의 에러는 2개의 인자들 간의*

1) 2인자 이상의 특정한 인자 수준의 조합에서 일어나는 효과를 교호작용이라 부른다.

*교호작용에 의해서 일어난다.* 따라서 고차의 교호작용이 존재하지 않는다는 가정은 대개 만족된다. 그러므로 본 논문에서도 데이타 부분에 대한 시험경우를 도출하는데 일부실시법을 적용할 수 있다.

일부실시법에서 직교배열표 (orthogonal array table)는 매우 유용하게 사용된다. 직교배열표는 요인간의 직교성을 이용하여 만들어 놓은 표로서 인자들의 쌍(pair)에 대한 모든 수준의 조합을 포함하는 시험스위트를 생성한다. 요인간에 직교성을 갖도록 실험계획하여 데이타를 구하면 같은 실험횟수라도 에러의 유무를 찾아내는 검출력이 더 좋은 시험을 할 수 있고, 더 정확하게 에러가 어디에 존재하는지를 추정 할 수 있는 것으로 밝혀져 있다 [7, 10].

Bellcore에서는 이러한 실험계획법에 기반하여 Automatic Efficient Test Generator (AETG) [6, 8, 9]를 개발하였다. AETG는 사용자 요구사항으로부터 시험경우를 자동적으로 생성하는 툴로 인자들의 모든 가능한 쌍을 포함하는 시험스위트를 만드는 시스템이다. AETG는 스크린 시험, 프로토콜 적합성 시험에 이용되었다. 특히 적합성 시험의 경우 ISDN의 호 거부 (Call Rejection)와 채널 협상 (Channel Negotiation)에 대해서 AETG를 적용하였다 [6].

2.2.1 직교배열표

인자의 수가 많을 경우에 실험횟수를 적게 할 수 있는 실험계획을 간단히 짤 수 있도록 만들어 놓은 표가 직교배열표이다 [10]. 직교배열표는 모든 가능한 시험경우를 다 수행하는 것보다 훨씬 적은 시험경우를 가지면서 인자들의 어떠한 쌍에 대해서도 모든 수준의 조합이 일어나는 시험경우를 생성하기 때문에 대부분의 인자 에러를 일으키는 교호작용을 커버한다.

직교배열표에는 2, 3, 4, 5 수준계 및 혼합수준계[2] 등이 있으나 일반적으로 2수준계와 3수준계가 널리 사용된다. 수준계마다 직교배열표를 만드는 방법은 다르다. 여기서는 2수준계 직교배열표에 대해서만 설명한다. $2^m$은 시험경우의 수이고 $2^m$-1은 열의 수이면서 배치가능한 인자의 최대수이다. m은 2 이상의 정수이다. 그러므로 직교배열표를 이용하면 최대 $2^m$-1개의 인자를 $2^m$개의 시험경우로 에러가 생길 여지가 있는 교호작용을 대부분 커버하면서 시험이 가능하다.

3개의 인자가 각각 2수준을 갖는 시험에 대해 시험경우를 구해보자. 수준값은 1과 2로 둔다. 전체 시험경우 (exhaustive test set)는 (1,1,1), (1,1,2), (1,2,1), (2,1,1), (1,2,2), (2,1,2), (2,2,1), (2,2,2)로 8개의 시험경우를 갖는다. 이는 3인자 교호작용을 커버하는 시험경우이다. 직교배열표를 이용하면 표 1에서 보는 바와 같이 (1,1,1), (1,2,2), (2,1,2), (2,2,1)로 4개의 시험경우를 갖는다. 이는 2인자 교호작용만을 커버하는 시험경우이다. 표 1은 m이 2인 경우로 최대 3 (즉, $2^2$-1)개의 인자를 4 (즉, $2^2$)개의 시험경우로 시험 가능한 직교배열표를 나타낸다. 직교배열표에서 어떤 인자를 어떤 열번호에 배치하느냐, 또는 각 인자에 대해서 어떤 수준을 1 또는 2에 배치하느냐는 임의로 결정한다. 따라서 이 경우에 직교배열표를 이용하면 시험경우는 50% 감소하게 되는데, 프로토콜 분야에서 3인자 이상의 교호작용은 거의 존재하지 않기 때문에 4개의 시험경우만으로 시험이 가능하다. 직교배열표를 이용하여 구한 시험경우는 전체의 시험경우와 마찬가지로 2인자 교호작용 즉, 인자의 쌍에 대한 모든 수준의 조합을 커버하게 된다. 표 1에서 보면, 임의의 두 열에 대해서 (1,1), (1,2), (2,1), (2,2)를 모두 포함하므로 2인자, 모든 쌍에 대한 조합을 100% 커버한다. 표 2는 전부 시험하는 방법과 직교배열표를 이용하는 방법에 대한 시험경우의 수와 커버리지 범위를 요약한다. 본 논문에서 커버리지의 범위는 시험 인자의 모든 쌍에 대한 조합 (all pairwise combination)의 백분율로 정의된다.

표 1 3인자, 2수준을 위한 직교배열표

| 실험번호 | 열번호 | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 1 | 1 | 1 |
| 2 | 1 | 2 | 2 |
| 3 | 2 | 1 | 2 |
| 4 | 2 | 2 | 1 |

표 2 시험경우의 수와 커버리지 범위

| 시험 예 | 시험경우의 수 | | 커버리지 범위 (%) | |
|---|---|---|---|---|
| | 전체 시험경우 | 직교배열표 이용 | 전체 시험경우 | 직교배열표 이용 |
| 3인자, 2수준의 시험 | 8 | 4 | 100 | 100 |

### 2.3 TCP 연결 설정에 대한 명세 사항

TCP의 동작은 연결 설정, 연결 종료, 데이타 전송 절차로 구성된다. 본 논문에서는 연결 설정 부분에 대해서만 데이타 부분에 대한 시험경우를 도출하므로 연결 설

2) 인자의 수준에 따라 2, 3, 4, 5 수준계로 나뉘며 혼합수준계는 각 인자의 수준수가 혼합된 직교배열표이다.

정에 대한 명세 사항을 기술한다. 그림 4는 TCP 헤더를 나타내고 있으며 괄호 안은 각 필드들이 차지하는 비트 수를 의미한다.

TCP는 연결 지향 프로토콜이므로 다른 쪽으로 데이타를 보내기 전에 그들 사이에 연결이 설정되어야 한다. 연결 설정을 위해 TCP는 세방향 핸드쉐이킹을 사용한다. 이 과정에는 TCP 패킷에 순서 번호 (sequence number : seq), 확인응답 번호 (acknowledgment number : ack), 윈도우 크기 (window size : win), 최대 세그먼트 크기 (maximum segment size : mss) 필드 등이 제어 필드 외에 포함되어 있다. TCP 연결 설정 절차에서 사용하는 필드는 그림 4에서 기울임체로 나타내고 있으며, 본 논문에서는 TCP 데이타 부분에 대한 시험경우를 생성하기 위해 이러한 인자들을 고려한다.

순서 번호와 확인응답 번호는 TCP 헤더 필드에서 32 비트를 차지하고 있다. 그러므로 순서 번호와 확인응답 번호는 0 ~ $2^{32}$-1의 값을 가질 수 있고 $2^{32}$-1에서 다시 0으로 돌아온다. 초기 순서 번호 (initial sequence number)는 시간이 지남에 따라서 변하기 때문에 연결할 때마다 다른 초기 순서 번호를 갖는다. 다시 연결 설정을 할 경우 순서 번호는 일반적으로 전 연결 설정으로부터 더 이상 존재하지 않는 숫자를 선택하게 되는데 약 4$\mu$s가 지난 후의 값을 초기 순서 번호로 선택하는 것이 좋다. 왜냐하면 순서 번호는 32 비트 카운터가 매 4$\mu$s마다 1씩 증가하기 때문이다 [11].

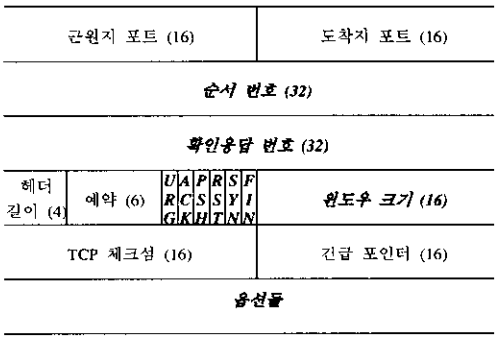


그림 4 TCP 헤더

윈도우 크기는 TCP 헤더의 16비트를 차지하며 윈도우 크기를 상대방에게 알려줌으로써 TCP 흐름 제어 (flow control)가 이루어진다. 윈도우 크기는 ACK 비트가 설정되었을 때만 의미가 있고 ACK을 보낼 때 윈도우 크기를 변화시킬 수 있다. 그러나 초기 연결 설정을 위해 패킷을 보낼 때 윈도우 크기도 함께 전송한다. 몇 가지 응용프로그램은 성능을 향상시키기 위해서 소켓 버퍼 크기를 변화시키지만 연결 설정 과정에는 데이타 교환이 없기 때문에 초기에 설정된 값이 변화하지 않는다. 4.2BSD에서 초기 송신 버퍼와 수신 버퍼는 각각 2048바이트이고 4.3BSD에서는 4096바이트이다. SunOS 4.1.3, BSD/386, SVR4는 4096을 기본값으로 사용하고 있고 Solaris 2.2, 4.4BSD, AIX3.2는 8192나 16384바이트를 사용한다 [12].

최대 세그먼트 크기는 TCP 헤더의 옵션 필드에서 16 비트를 차지하며, 연결을 설정하는 동안에만 지정된다. 최대 세그먼트 크기는 보내는 쪽에서 설정하게 되는데 보내는 쪽은 이 값보다 더 큰 TCP 세그먼트를 받기를 원하지 않는다. 최대 세그먼트 크기를 설정하는 이유는 패킷 분할 (fragmentation)을 피하고자 함이다. Ethernet에서 최대 세그먼트 크기는 1460바이트이다. BSD/386과 SVR4를 포함하는 연결에 대해서 최대 세그먼트 크기는 1024인데 이는 BSD 구현물들은 최대 세그먼트 크기로 512의 배수를 요구하기 때문이다. SunOS 4.1.3, Solaris 2.2, AIX 3.2.2와 같은 시스템에서 종단간이 Ethernet이라면 최대 세그먼트 크기는 1460이고 SLIP 링크에서는 256이다 [12].

그림 5는 TCP 연결 설정에 따른 세그먼트의 tcpdump [13] 결과이다.

```
1 svr4.1037 > bsdi.discard: S 1415531521:1415531521(0) win 4096 <mss 1024>
2 bsdi.discard > svr4.1037: S 1823083521: 1823083521(0) ack 1415531522 win 4096 <mss 1024>
3 svr4.1037 > bsdi.discard: . ack 1823083521 win 4096
```

그림 5 TCP 연결 설정에 대한 tcpdump 결과

그림 5에 있는 3개의 TCP 세그먼트는 연결 설정에 대한 TCP 헤더만을 담고 있다. 데이타 (payload)는 교환되지 않는다. TCP 세그먼트에 대해 각각의 결과는 " 근원지 > 도착지 : 플래그" 형식으로 표현된다. 플래그는 6비트의 제어 필드를 나타내고 1415531521: 1415531521(0)는 순서 번호가 1415531521이고 데이타 바이트가 0임을 의미한다. ack 다음의 숫자는 확인응답 번호를 나타내고 win다음의 숫자는 윈도우 크기, mss 다음의 숫자는 가장 일반적인 옵션 필드인 최대 세그먼트 크기를 나타낸다.

## 3. TCP 데이타 부분에 대한 상호운용성 시험경우 생성

이 장에서는 기존 연구 [1]에서 도출한 TCP 상호운용성 시험경우에 데이타 부분을 포함하는 시험경우를

도출하는 과정을 살펴본다. 그림 6은 TCP 데이타 부분에 대한 상호운용성 시험경우 도출 과정을 나타낸다. 논문 [1]에서 제어 부분에 대한 시험경우는 TCP FSM을 구현한 프로그램에 입력으로 넣어 도출하였다. 이렇게 도출된 제어 부분에 대한 시험경우는 불가능한 동작 시퀀스 (sequence)를 배제하는 효과가 있다. 본 연구의 방법에서 데이타 부분에 대한 시험경우는 제어 부분에 대한 시험경우를 근간으로 하여 도출하게 된다. 3.1절에서는 제어부분에 대한 시험경우를 구하기 위해 가정을 세우고, 3.2절에서는 이들 가정을 고려하여 시험경우를 구한다. 3.3절에서는 3.2절에서 구한 시험경우에 실험계획법의 직교배열표를 이용하여 다시 시험경우를 구한다. 그리고 3.4절에서는 각 단계에서 도출된 시험경우의 수를 구하여 비교해본다.

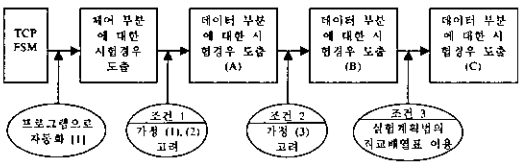


그림 6 데이타 부분에 대한 시험경우 도출 과정

### 3.1 가정

제어 부분에 대한 시험경우에 데이타 부분을 포함한 시험경우를 도출하기 위해서는 시험 목적에 대한 가정이 필요하다.

(1) 두 개의 TCP에 있는 각각의 순서 번호, 윈도우 크기, 최대 세그먼트 크기를 나타내는 총 6개의 인자를 고려한다.

그림 7은 일반적인 TCP 연결 설정 절차를 나타내는

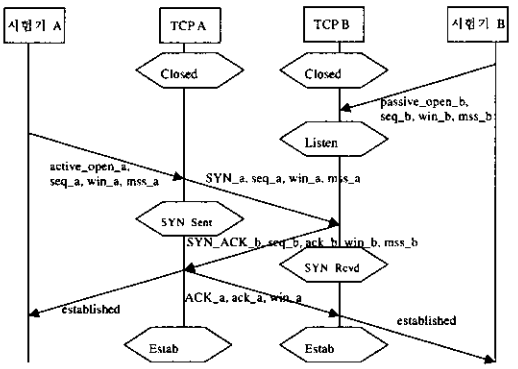


그림 7 TCP 연결 설정을 나타내는 Message Sequence Chart

것으로 각 세그먼트마다 몇 개의 인자들을 가지고 있다. 확인응답 번호는 초기값을 지정해 줄 필요가 없다. 상대편 패킷의 순서 번호를 받으면 확인응답 번호로 상대편 패킷의 순서 번호에 1을 더하여 다음에 자신이 받을 패킷의 번호를 보내기 때문이다. 이렇게 각 TCP마다 3개의 인자에 초기값을 지정해 주면 확인응답 번호나 다음 순서 번호, 다음 윈도우 크기는 각 인자의 초기값으로 인해 결정되기 때문에 초기값만을 시험기에서 정해주면 된다.

(2) 연결 설정을 초기화 할 때는 그림 7에서 보는 것처럼 양쪽 시험기에서 각각 TCP A와 TCP B로 각 인자들의 초기값을 전달한다고 가정한다. active_open이나 passive_open, send_data라는 메시지가 TCP 응용프로그램에서 주어지게 되면 TCP 패킷의 3개의 인자들의 값이 초기화되고 패킷 교환이 일어날 때 상대편 TCP로 전달된다.

(3) 모든 인자의 수준을 2 수준으로 설정하고 각 인자의 수준은 표 3과 같이 선택한다.

각 인자가 가질 수 있는 값의 범위는 너무 크기 때문에 모든 값을 선택하는 것은 불가능하다. 따라서 수준의 수를 적게 하면서도 샘플링 커버리지 (sampling coverage)를 더 높이도록 수준을 선택해야 한다. 책 [7]에 의하면 인자의 수준을 택하는 방법은 다음의 원칙에 의해 행한다. 첫째, 수준을 선택할 때에는 수준의 최대치와 최소치를 취해주는 것이 좋으며 그 중간 수준은 수준영역을 등간격으로 나누어 잡아주는 것이 좋다. 둘째, 현재 사용되고 있는 인자의 수준은 포함시키는 것이 좋으며 최적이라고 예상되는 수준도 포함시킨다. 셋째, 인자의 수준수는 보통 2~5 수준이 적절하며 많아도 6 수준을 넘지 않도록 한다. 본 논문에서는 TCP A와 B의 순서 번호로 수준영역을 등간격으로 나누어 잡고 수준의 최대치와 최소치가 포함되도록 선택하였다. 윈도우 크기와 최대 세그먼트 크기는 현재 시스템에서 사용되고 있는 값을 수준으로 포함시켰다. 이러한 값은 PICS/PIXIT (Protocol Implementation Conformance Statement / Protocol Implementation eXtra Information for Testing)을 통하여 TCP 구현자가 시험자에게 주게 된다.

표 3 인자들의 수준

| 인자 / 수준 | TCP A의 seq | TCP A의 win | TCP A의 mss | TCP B의 seq | TCP B의 win | TCP B의 mss |
|---|---|---|---|---|---|---|
| 1 | 282083621 | 2048 | 1460 | 1415531521 | 4096 | 1024 |
| 2 | 0 | 8192 | 256 | 429496735 | 16384 | 256 |

### 3.2 가정에 의한 시험경우 도출

데이타 부분에 대한 시험경우를 도출하기 위해서 먼저 가정 (1)과 (2)를 고려 (조건 1)하면 부록 1에 있는 시험경우 결과는 데이타 부분에 대한 시험경우를 도출할 때 두 가지 형태로 도출된다. 먼저 첫 번째 형태의 예를 들어보자. 부록 1의 시험경우 <1>은 데이타 부분에 대한 시험경우가 다음과 같은 형태로 도출된다. 그림 7은 다음의 시험경우에 대한 시나리오이다.

(Closed,Listen) — active_open_a/[ <,(i_SYN,seq,win, mss),,(i_SYN_ACK,seq,ack,win, mss)>, <established,(i_ACK, ack,win),established,> ] → (Estab,Estab)

초기 안정 상태가 (Closed,Listen)이므로 TCP B는 이미 시험기 B로부터 passive_open이라는 메시지를 받고 자신의 3개 인자의 초기값을 가지고 있다. TCP A는 시험기 A로부터 active_open 메시지를 받고 3개 인자의 초기값을 가지게 되어 상대편 TCP에 패킷을 보내게 된다. 따라서 양쪽 TCP 모두 3개의 인자를 사용하기 때문에 결국 위의 예는 6개의 인자가 추가된 시험경우가 된다. 이와 같이 데이타 부분에 대한 시험경우를 도출할 때 6개의 인자를 사용하는 시험경우는 부록 1의 결과 중 <1>에서 <8>까지 8개이다.

다음, 두 번째 형태의 예를 들어보자. 부록 1의 시험경우 <9>는 데이타 부분에 대한 시험경우가 다음과 같은 형태로 도출된다. 그림 8은 다음의 시험경우에 대한 시나리오이다.

(Closed,Closed) — active_open_a/[ <,(i_SYN,seq,win, mss),,> ] → (SYN_Sent,Closed)

TCP B는 시험기 B로부터 passive_open 메시지를 받지 못하여 TCP A의 패킷을 받을 준비가 되지 않았으므로 Closed 상태에 머물게 된다. 따라서 위의 시험경우는 TCP A의 3개의 인자만이 사용된다. 이와 같이 데이타 부분에 대한 시험경우를 도출할 때 3개의 인자를 사용하는 시험경우는 부록 1의 결과 중 <9>에서 <12>까지 4개이다.

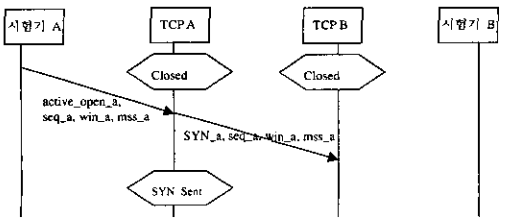


그림 8 한쪽 TCP 인자만을 사용하는 경우의 Message Sequence Chart

각 인자가 가질 수 있는 모든 값을 수준으로 선택하여 시험경우의 수를 구해보자. TCP의 순서 번호 필드는 TCP 헤더에서 32 비트를 차지하기 때문에 총 가능한 수준수는 $2^{32}$이고, 윈도우 크기와 최대 세그먼트 크기는 16비트여서 수준수는 $2^{16}$이다. 따라서 부록 1에서 6개의 인자를 모두 사용하는 8개의 시험경우는 각각 $2^{128}$ (즉, $2^{32}\times2^{32}\times2^{16}\times2^{16}\times2^{16}\times2^{16}$)개의 데이타 부분에 대한 시험경우로 확장되고 3개의 인자를 사용하는 4개의 시험경우는 각각 $2^{64}$ (즉, $2^{32}\times2^{16}\times2^{16}$)개의 시험경우로 확장된다. 따라서 데이타 부분에 대한 전체 시험경우는 $2^{131}+2^{66}$ (즉, $8\times2^{32}\times2^{3}\times2^{2}2^{16}\times2^{16}\times2^{16}\times2^{16}+4\times2^{32}\times2^{16}\times2^{16}$)개가 된다.

이제, 가정 (3)도 고려 (조건 2)하여 데이타 부분에 대한 시험경우의 수를 구해보자. 부록 1에서 데이타 부분에 대한 시험경우를 도출할 때 6개의 인자를 모두 사용하는 8개의 시험경우는 각각 64 (즉, $2^6$) 개의 시험경우로 확장되고, 3개의 인자를 사용하는 나머지 4개의 시험경우는 각각 8 (즉, $2^3$)개의 시험경우로 확장된다. 따라서 전체 시험경우는 544 (즉, $8\times2^6+4\times2^3$)개이다.

### 3.3 직교배열표를 이용한 시험경우 도출

3.2절에서 구한 시험경우의 수는 실험계획법의 직교배열표를 이용 (조건 3)하여 더 줄어들 수 있다. 표 4는 2.2절에서 언급한 m이 3인 경우의 2수준계 직교배열표로서 2수준의 실험을 위한 6인자 배치를 나타낸 것이다. 이는 2 수준을 가진 6개의 인자의 모든 조합을 고려하는 64 (즉, $2^6$)회의 시험을 하지 않고 8회만으로 시험이 가능하다는 것을 의미한다. 6인자를 배치하는 경우에는 직교배열표의 7개의 열 중에서 6개의 열을 임의로 택하여 배치하면 된다. 열번호 1, 2, 3, 4, 5, 6을 택하여 배치하면 열번호 7은 실험에 대한 오차 (e)가 되지만 상호운용성 시험에서는 의미가 없다. 각 인자에 대해서 어떤

표 4 6인자, 2수준을 위한 직교배열표

| 실험 번호 | 열번호 | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 |
| 3 | 1 | 2 | 2 | 1 | 1 | 2 | 2 |
| 4 | 1 | 2 | 2 | 2 | 2 | 1 | 1 |
| 5 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| 6 | 2 | 1 | 2 | 2 | 1 | 2 | 1 |
| 7 | 2 | 2 | 1 | 1 | 2 | 2 | 1 |
| 8 | 2 | 2 | 1 | 2 | 1 | 1 | 2 |
| 배치 | TCP A 의 seq | TCP A 의 win | TCP A 의 mss | TCP B 의 seq | TCP B 의 win | TCP B 의 mss | e (오차) |

수준을 1또는 2에 배치하느냐는 임의로 결정하므로 여기서는 표 3의 배치를 따른다.

부록 1에 있는 8개의 시험경우는 데이타 부분에 대한 시험경우를 도출할 때 6개의 인자를 사용하기 때문에 표 4의 직교배열표를 이용하고, 나머지 4개의 시험경우는 3개의 인자를 사용하기 때문에 표 1의 직교배열표를 이용하여 데이타 부분의 인자를 배치한다. 표 1의 직교배열표를 이용하는 경우 1, 2, 3열을 각각 seq, win, mss로 배치하였다. 따라서 데이타 부분에 대한 전체 시험경우 수는 80 (즉, 8×8+4×4)개이다. 부록 2는 부록 1의 시험경우 중 <1>과 <9>에 대해서만 데이타 부분을 고려한 시험경우로서, 각각 8개, 4개씩 도출된다. 다음은 부록 1의 <1> 시험경우에 데이타 부분을 고려하여 생성한 시험경우 중 하나로 부록 2의 <1> 시험경우이다. 부록 2의 <1> 시험경우는 표 4의 실험번호 1에 의하여 각 인자의 수준이 배치된 시험경우이다.

(Closed,Listen) — active_open_a/[ <,(i_SYN,seq=2823083521, win=2048,mss=1460),, (i_SYN_ACK,seq= 1415531521,ack= 2823083522,win=4096,mss=1024)>, <established, (i_ACK_a,ack= 1415531522,win=2048),established,> ] → (Estab,Estab)

TCP A는 연결 설정을 위해 초기 순서 번호를 2823083521, 윈도우 크기를 2048, 최대 세그먼트 크기를 1460으로 갖는 SYN 세그먼트를 TCP B로 보냈다. TCP B는 초기 순서 번호를 1415531521로, 윈도우 크기를 4096, mss를 1024로 갖는 SYN 세그먼트를 보내면서, 받은 SYN에 대해서 상대의 초기 순서 번호에 1을 더하여 ACK으로 승인하였다. TCP A는 TCP B의 초기 순서 번호에 1을 더하여 상대로부터 받은 SYN을 승인하여 세방향 핸드쉐이킹이 이루어졌다.

### 3.4 평가

표 5는 데이타 부분에 대한 시험경우를 도출하기 위한 그림 6의 3가지 조건을 차례로 고려하면서 구한 시험경우의 수를 나타낸다. 그림 1의 조건 1을 고려할 경우에 데이타 부분에 대한 전체 시험경우의 수를 구하면 $2^{131}+2^{66}$개이고, 조건 2까지 고려하면 544개로 감소된다. 여기에 직교배열표를 이용하는 조건 3을 추가하면 시험에 대한 검출력을 유지하면서 80개의 시험경우로 줄일 수 있다. 이는 가정의 모든 항목을 고려하여 도출한 544개의 시험경우 (B)에 비해서 85% 감소한 것이고, (A)에 비해서는 99.999%보다 더 감소한 것이다. 또한 프로토콜 분야에서 대부분의 인자 에러는 2인자의 교호작용으로 인해 일어나기 때문에 이를 모두 커버하는 80개의 시험경우는 544개의 시험경우를 가지고 시험을 하는 것과 거의 같은 효과를 갖는다.

표 5 데이타에 대한 시험경우를 도출하는 과정에서 시험경우 수 비교

| 그림 6의 표기 | 데이타 부분에 대한 시험경우 도출을 위한 조건 | 시험경우의 수 | 비고 |
|---|---|---|---|
| (A) | 조건 1. 가정 (1), (2)를 고려한 경우 | 2131+266 | (C)는 (A)와 비교하여 99.999% 이상 감소하고 (B)와 비교하여 85% 감소 |
| (B) | 조건 2. 가정 (3)을 고려한 경우 | 544 | |
| (C) | 조건 3. 직교배열표를 이용한 경우 | 80 | |

## 4. 결론 및 향후 연구

본 논문은 제어 부분으로만 구성된 시험경우에 데이타 부분을 고려하여 TCP의 연결 설정 절차에 대한 시험경우를 도출하는 방법을 개발하였다. 데이타 부분에 대한 시험경우를 도출하기 위해서 시험에 대한 가정을 세웠고 실험계획법을 이용하여 시험에 대한 검출력을 유지하면서 시험경우의 수를 최소화하였다. 이러한 시험경우는 80개로, 가정의 모든 항목을 고려하여 도출한 544개의 시험경우에 비해서 85%가 감소한 것이고, 인자의 수준수를 제한하는 가정을 고려하지 않은 $2^{131}+2^{66}$개의 시험경우에 비해서 99.999%보다 더 감소한 것이다. 이렇게 시험경우를 감소시키게 되면 상호운용성 시험에 드는 시간과 비용은 훨씬 줄어들게 된다.

TCP 데이타 부분에 대한 상호운용성 시험경우를 도출하기 위한 방법은 연결 설정 부분에만 적용되었을 뿐, 아직 데이타 전송 부분과 연결 종료 부분에는 적용되지 않았다. 이 부분에 대한 시험경우 도출 방법 연구와 TCP 데이타 부분에 대한 시험경우를 생성하는 알고리즘을 정립하는 연구가 요구된다. 그리고 이를 자동화하여 본 논문에서 수작업으로 도출한 시험경우와 결과를 비교 및 분석하여 알고리즘의 정확성을 파악하고, ATM/B-ISDN 신호 프로토콜에 적용하여 이 알고리즘의 일반성을 보일 계획이다.

## 참 고 문 헌

[1] S. Seol, M. Kim, S. Kang, and Y. Park, "Interoperability Test Suite Derivation for the TCP," IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), October 5-8, Beijing, China, 1999.

[2] S. Kang and M. Kim, "Interoperability Test Suite Derivation for Symmetric Communication Protocols," IFIP Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols(FORTE X) and Protocol Specification Testing and Verification (PSTV XVII), pp. 57-72, November 1997.
[3] J. Shin and S. Kang, "Interoperability Test Suite Derivation for the ATM/B-ISDN Signaling Protocol," Testing of Communicating Systems, Vol. 11, Kluwer Academic Publishers, pp. 313-330, 1998.
[4] ITU-T X.290 Series, Conformance Testing *Methodology and Framework*, 1994.
[5] ISO/IEC/9646, OSI Conformance Testing Methodology and Framework Parts 1-7, 1994.
[6] K. Burroughs, A. Jain, and R. L. Erichson, "Improved Quality of Protocol Testing Through Techniques of Experimental Design," Supercomm/ICC '94, 1994.
[7] Douglas C. Montgomery, Design and Analysis of Experiments, 4th Ed., Wiley, 1997.
[8] D. M. Cohen, S. R. Dalal, A. Kajla, and G. C. Patton, "The Automatic Efficient Test Generator (AETG) System", Proc. 5th Int'l Symp. Software Reliability Eng., IEEE CS Press, Los Alamitos, Calif., pp. 303-309, 1994.
[9] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton, "The Combinatorial Design Approach to Automatic Test Generation," IEEE Software Volume: 13, Issue: 5, pp. 83-88, Sept. 1996.
[10] G. S. Peace, Taguchi Methods: A Hands-On Approach, Addison-Wesley, 1993.
[11] J. B. Postel, "Transmission Control Protocol," RFC 793, Sept. 1981.
[12] W. Richard Stevens, TCP/IP Illustrated, Volume1: The Protocols, Addison-Wesley, 1995.
[13] V. Jacobson, C. Leres, and S. McCanne, The Tcpdump Manual Page, Lawrence Berkeley National Laboratory, Berkeley, CA., June 1997.
[14] B. Sarikaya, G. V. Bochmann, and E. Cerny, "A Test Design Methodology for Protocol Testing," IEEE Transaction On Software Engineering, VOL. SE-13, NO. 5, May 1987.

## 부 록

1. TCP 연결 설정 부분과 관련된 제어 부분에 대한 상호운용성 시험경우.

<1> (Closed,Listen) — active_open_a/[ <,i_SYN,,i_SYN_ACK>, <established,i_ACK, established,> ] → (Estab,Estab)

<2> (Listen,Closed) — active_open_b/[ <,i_SYN_ACK,,i_SYN>, <established,,established, i_ACK > ] → (Estab,Estab)

<3> (Closed,SYN_Sent) — active_open_a/[ <,i_SYN,,i_SYN_ACK>, <established,i_ACK, established,> ] → (Estab,Estab)

<4> (SYN_Sent,Closed) — active_open_b/[ <,i_SYN_ACK,,i_SYN>, <established,, established,i_ACK> ] → (Estab,Estab)

<5> (Listen,Listen) — send_data_a/[ <,i_SYN,,i_SYN_ACK>, <established,i_ACK, established,> ] → (Estab,Estab)

<6> (Listen,Listen) — send_data_b/[ <,i_SYN_ACK,,i_SYN>, <established,,established, i_ACK > ] → (Estab,Estab)

<7> (Listen,SYN_Sent) — send_data_a/[ <,i_SYN,,i_SYN_ACK>, <established,i_ACK, established,> ] → (Estab,Estab)

<8> (SYN_Sent,Listen) — send_data_b/[ <,i_SYN_ACK,,i_SYN>, <established,, established,i_ACK > ] → (Estab,Estab)

<9> (Closed,Closed) — active_open_a/[ <,i_SYN,,> ] → (SYN_Sent,Closed)

<10> (Closed,Closed) — active_open_b/[ <,,,i_SYN> ] → (Closed,SYN_Sent)

<11> (Listen,Closed) — send_data_a/[ <,i_SYN,,> ] → (SYN_Sent,Closed)

<12> (Closed,Listen) — send_data_b/[ <,,,i_SYN> ] → (Closed,SYN_Sent)

2. 부록 1의 <1>과 <9>의 시험경우에 대해서 데이타 부분이 적용된 시험경우.

<1> (Closed,Listen) — active_open_a/[ <,(i_SYN,seq=2823083521,win=2048,mss=1460),, (i_SYN_ACK,seq=1415531521,ack=2823083522,win=4096,mss=1024)>, <established, (i_ACK,ack=1415531522,win=2048),established,> ] → (Estab,Estab)

<2> (Closed,Listen) — active_open_a/[ <,(i_SYN,seq=2823083521,win=2048,mss=1460),, (i_SYN_ACK,seq=4294967295,ack=2823083522,win=16384,mss=256)>, <established, (i_ACK,ack=0,win=2048),established,> ] → (Estab,Estab)

<3> (Closed,Listen) — active_open_a/[

<,(i_SYN,seq=2823083521,win=8192,mss=256),,
(i_SYN_ACK,seq=1415531521,ack=2823083522,win=4096,mss=256)>, <established,
(i_ACK,ack=1415531522,win=8192),established,> ] → (Estab,Estab)

<4> (Closed,Listen) — active_open_a/[
<,(i_SYN,seq=2823083521,win=8192,mss=256),,
(i_SYN_ACK,seq=4294967295,ack=2823083522,win=16384,mss=1024)>, <established,
(i_ACK,ack=0,win=8192),established,> ] → (Estab,Estab)

<5> (Closed,Listen) — active_open_a/[
<,(i_SYN,seq=0,win=2048,mss=256),,(i_SYN_ACK,
seq=1415531521,ack=1,win=16384,mss=1024)>,
<established,(i_ACK,ack=1415531522,
win=2048),established,> ] → (Estab,Estab)

<6> (Closed,Listen) — active_open_a/[
<,(i_SYN,seq=0,win=2048,mss=256),,(i_SYN_ACK,
seq=4294967295,ack=1,win=4096,mss=256)>,
<established,(i_ACK,ack=0,win=2048),
established,> ] → (Estab,Estab)

<7> (Closed,Listen) — active_open_a/[
<,(i_SYN,seq=0,win=8192,mss=1460),,
(i_SYN_ACK,seq=1415531521,ack=1,win=16384,mss=256)>,
<established,(i_ACK,
ack=1415531522,win=8192),established,> ] → (Estab,Estab)

<8> (Closed,Listen) — active_open_a/[
<,(i_SYN,seq=0,win=8192,mss=1460),,
(i_SYN_ACK,seq=4294967295,ack=1,win=4096,mss=1024)>,
<established,(i_ACK,
ack=0,win=8192),established,> ] → (Estab,Estab)

<9> (Closed,Closed) — active_open_a/[
<,(i_SYN,seq=2823083521,win=2048,mss=1460),,>
] → (SYN_Sent,Closed)

<10> (Closed,Closed) — active_open_a/[
<,(i_SYN,seq=2823083521,win=8192,mss=256),,> ]
→ (SYN_Sent,Closed)

<11> (Closed,Closed) — active_open_a/[
<,(i_SYN,seq=0,win=2048,mss=256),,> ] →
(SYN_Sent,Closed)

<12> (Closed,Closed) — active_open_a/[
<,(i_SYN,seq=0,win=8192,mss=1460),,> ] →
(SYN_Sent,Closed)


유 지 원

1998년 8월 아주대학교 산업공학과 졸업(학사). 1999년 3월 ~ 현재 한국정보통신대학원대학교 공학부 석사과정. 1999년 3월 ~ 1999년 12월 한국통신 통신망 연구소 시험기술연구실 위촉연구원. 관심분야는 프로토콜 시험, 차세대인터넷, 이동 컴퓨팅.

김 명 철

1982년 아주대학교 전자공학과 졸업(학사). 1984년 KAIST 전산학과 졸업(석사). 1992년 Univ. of Brithish Columbia(박사). 1984년 ~ 1997년 한국통신 연구개발본부 표준연구단 실장. Co-chair of 10th International Workshop on Testing of Communication Systems, 1997년 PTS-SIG chair of Asia Oceania Workshop. 1997년 ~ 현재 한국정보통신대학원 교수. 관심분야는 통신망 및 멀티미디어 프로토콜 공학

설 순 욱

1998년 2월 한국기술교육대학교 정보통신공학과 졸업(학사). 2000년 2월 한국정보통신대학원대학교(공학석사). 1998년 ~ 1999년 한국전자통신연구원 네트워크 장비시험센터 위촉연구원. 2000년 3월 ~ 현재 한국정보통신대학원대학교 박사과정. 관심분야는 프로토콜 시험, 이동 컴퓨팅, 차세대인터넷.

강 성 원

1982년 서울대학교 사회과학대학 졸업. 1985년 ~ 1986년 미국 University of Iowa 전산학부. 1989년 미국 University of Iowa 전산학 석사. 1992년 미국 University of Iowa 전산학 박사. 1993년 한양대학교 전자계산학과 강사. 1995년 ~ 1996년 미국 국립표준기술연구소(NIST) 객원연구원. 1997년 IWTCS '97 국제학술대회 공동의장. 1993년 ~ 현재 한국통신 연구개발본부 선임연구원(프로토콜시험부장). 관심분야는 통신프로토콜 시험자동화, 상호운용성 시험이론, 프로그램 최적화, 선언적 프로그래밍 언어

이 영 희

정보과학회논문지 : 정보통신
제 28 권 제 1 호 참조

이 근 구
1982년 연세대학교 전자공학과(학사). 1985년 연세대학교 전자공학과(석사). 1994년 ~ 1995년 미국 NIST 객원연구원 근무. 1994년 ~ 1997년 미국 해외사무소(워싱턴) 파견 근무. 1984년 ~ 현재 전자통신연구원 상호운용성시험팀 팀장/선임연구원. 관심분야는 신호 프로토콜, 정보통신 상호운용성 시험, 서비스 품질 및 망성능