JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 25 No. 9, pp. 193-201, September 2020
https://doi.org/10.9708/jksci.2020.25.09.193

Development of Simulation App for Understanding Test-and-Set Algorithms that Multi Learner Can Use Simultaneously

Kyong-ho Lee*

*Professor, Dept. of Information Communication and Software, Halla University, Kangwon, Korea

[Abstract]

In this study, we developed a simulation app that performs the Test-and-Set algorithm. The test-and-set algorithm is a highly difficult algorithm, so this simulation app was created for learners who have difficulty understanding it. Learners who want to understand the Test-and-Set algorithm gather to form a team, and use this simulation app to discuss and practice, and these teams can practice at the same time. The test-and-set, which is assumed to be a machine language, is not interrupted by using a queue, and it can be seen that the configured simulation app performs well in all three conditions of 'mutual exclusion', 'progress', and 'bounded waiting' that must be solved in the critical area problem.

▸ **Key words**: Algorithm, Concurrency, Critical Section, Test-and-Set

[요 약]

본 연구에서는 Test-and-Set 알고리즘을 수행하는 시뮬레이션 앱을 개발하였다. Test-and-Set 알고리즘은 고난이도 알고리즘이라 이를 이해하기 어려워하는 학습자들을 위하여 이 시뮬레이션 앱을 만들었다. 본 시뮬레이션 앱을 이용하여 Test-and-Set 알고리즘의 이해를 원하는 학습자들이 모여 팀을 이루고 논의하며 실습을 할 수 있도록 하였으며, 이런 여러 팀들이 동시에 실습할 수 있도록 만들었다. 구성한 시뮬레이션 앱에서 기계어 명령어라고 가정한 Test-and-Set 명령어를 큐를 이용하여 인터럽트 받지 않게 구성하였으며, 임계영역 문제에서 해결해야 하는 '상호배제', '진행', '제한된 대기'의 3조건이 모두 잘 수행됨을 볼 수 있었다.

▸ **주제어**: 알고리즘, 병행처리, 임계영역, Test-and-Set

• First Author: Kyong-ho Lee, Corresponding Author: Kyong-ho Lee
*Kyong-ho Lee (khlee@halla.ac.kr), Dept. of Information Communication and Software, Halla University
• Received: 2020. 08. 19, Revised: 2020. 09. 08, Accepted: 2020. 09. 08.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

한국의 스마트폰 보급률은 2020년 현재 95%에 달한다[1]. 대학생 보급률은 100%라고 말해도 문제가 없다. 우리나라는 네트워크가 잘 발달되어 있어 언제 어디서나 기기를 사용하는데 문제가 없다. 젊은이들에게 스마트폰은 장난감이자, 사람과 사람을 연결하는 환경이고, 책도 되고, 길도 안내해 주며, 모든 정보를 알아내는 친숙한 도구이다. 최근에 학습자들이 학습 내용을 잘 이해할 수 있도록 돕는 방법 중 하나로 수업 중에 앱을 많이 이용하고 있다. 수업과 업무에 도움이 될 앱도 많아서 미술과 테크놀러지가 만나서 멋있는 미술 도구가 되기도 하며, 음악 도구가 되기도 하고, 과학 등 수업에 내용을 시각적으로 보여주기도 하고, 영화 제작도 하며, 보건과 예방 교육, 인문학, 공학, 농업 교육 등 유치원 교육에서 대학교육 또 다양한 사회의 교육까지 다양한 과목에서 다양한 도움을 주고 있다[2-5].

본 연구에서는 운영체제의 Test-and-Set 알고리즘을 대상으로 의도를 반영하며 작동하는 시뮬레이션 앱을 구현하였다. 운영체제의 임계역역 문제를 해결하는 알고리즘들은 고난이로 학생들이 매우 어려워한다. 컴퓨터 자원을 공유하면서 병행 수행하는 다중 프로그래밍에서도 모든 프로세스는 바르게 작동되어야 한다. 각 프로세스에 가한 동일한 입력에 대해 프로세스들의 다양한 수행 순서에도 처리결과는 같아야 한다. 이를 결정성이라고 한다. Test-and-Set 알고리즘은 운영체제의 병행 처리를 위해 개발된 알고리즘으로 결정성을 유지하면서도 임계역역 문제를 매우 잘 해결한 알고리즘이다. 그런데 고난이므로 많은 학습자들이 이해에 어려움을 겪고 있다. 또 이해를 했더라도 평가를 통해 확인해 보면 수업 시간에 이해한 내용들을 기억하지 못하는 상황을 볼 수 있다. 이 문제를 해결하기 위해 시뮬레이션 앱을 개발했다. 인간의 기억은 보고-듣기만 할 때 보다, 보고-듣고-실습할 때 더 많은 내용을 기억한다는 이론을 배경으로 하였다. 또 알고리즘의 작동을 이해시키기 위한 시뮬레이터이므로 실제 작동을 연상시키는 환경에서 작동하며, 학습자가 자신의 의도를 반영하여 작동시키며, 작동 명령들의 변화와 다른 프로세스의 수행을 포함하여 각 명령에 따른 변수들 값의 변화를 볼 수 있다. 특히 본 앱이 설치된 각 폰은 하나의 프로세스이지만 Test-and-Set 알고리즘은 다수의 프로세스가 하나의 운영체제 안에서 모둠을 이루어 변수를 공유하게 해야 하므로 외부 데이터베이스를 이용하여 공유 변수를 구성하고 모둠 안의 앱 중 하나가 통제하도록 하였다. 학습자 모둠이 여럿일 때도 모둠별로 데이터베이스 영역이 분할되도록 구성하여 서로 간섭 없이 시뮬레이션을 할 수 있도록 하였다.

II. Preliminaries

본 연구의 시작은 수업하면서 정적 그림과 강의만으로 부족한 내용들을 바탕으로 학습자들에게 흥미와 이해 증진, 협력과 사고력 향상에 도움을 주기 위하여 교수자로서 인지한 필요성을 바탕으로 개발하고 이용하고자 한 것이다. 앱 선택 이유는 모든 학습자가 소지하고 있는 스마트폰 환경을 생각한 것이다. 연구자가 2년간 정적 그림과 강의로 Test-and-Set을 학습자들에게 전달한 효과를 평가를 이용하여 확인한 결과는 그림과 같다. y축은 빈도수이며, x축은 답을 근거로 본 연구자가 분류한 것이다. 100% 이해한 사람이 대략 10% 정도 되며, 100% 이해한 사람과 80% 이상 이해한 사람을 하나의 데이터로 잡으면 이 그래프는 U자 형태의 아주 독특한 분포를 보인다.

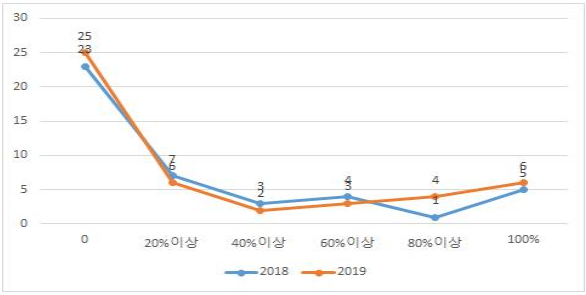

Fig. 1. Analysis table of effects delivered to learners only with static pictures and lectures of Test-and-Set algorithm

교육용 앱은 학습형태와 상호작용 방식에 따라 개인교수용, 반복학습형, 시뮬레이션형, 게임형, 자료제시형, 평가형, 도구형, 문제해결형으로 분류한다. 개인교수형은 새로운 개념이나 지식을 가르치기 위해 설명이나 안내를 하는 유형이며, 반복학습형은 일종의 보충이나 심화학습을 위해 반복적으로 지식을 확인하거나 기능을 연습하는 유형이며, 시뮬레이션은 실제와 유사한 환경을 제공하여 학습자가 연습할 수 있도록 하는 유형이며, 게임형은 목표, 규칙, 경쟁, 흥미, 도전, 호기심 등의 게임적인 요소를 첨가하여 학습을 하도록 설계된 유형이다. 자료제시형은 콘텐츠 안에 많은 분량의 자료를 저장하여 필요한 자료를 찾아보며 학습하도록 한 유형이며, 평가형은 교수-학습 결과의 평가를 주된 내용으로 만든 유형이며, 도구형은 학습을 촉진시켜 주기위한 도구적 성격이며, 문제해결형은 학습자가 고차원적인 사고 기능을 활용하여 비구조화된 문제를 해결하도록 하는 유형이다[6-7]. 앱 스토어를 통해 교육용 앱들에 관하여 분석해 보면 대부분의 교육용 앱은 한자 학습이나 영어 단어 암기 등과 같은 반복 학습형과 선생님이

설명해주는 동영상이나 애니메이션 등과 같은 전통적인 튜토리얼 형태의 개인교수형이 많다. 조연정 등의 연구에서도 튜토리얼 형태와 개인 교수형 앱들이 교육용에서 대부분을 차지하고 있다[7]. 본 연구자가 확인한 바로 대부분 교육용 앱은 보편적 활용을 위해 만든 것이며, 교수 현장에서 학습자의 필요성을 파악하고 반영한 것은 아니라고 판단되며, 특히 시뮬레이션형의 교육용 앱은 천체 관측용 앱 정도를 발견했을 뿐이다.

본 연구에서 개발한 Test-and-Set 시뮬레이터 앱을 구성하기 위하여 사용한 환경은 앱 인벤터(App Inventor)와 파이어베이스(Firebase)이다. 앱 인벤터는 2010년 구글에서 만들어 공개되었고 현재는 MIT에서 관리되고 있는 웹 비주얼 프로그래밍 툴이다. 컴퓨터 프로그래밍을 처음 접하는 사람들도 안드로이드 운영 체제용 앱을 쉽게 만들 수 있게 해준다. 시각 객체들을 드래그-앤-드롭하여 프로그램들을 만들 수 있다[8]. 그러나 간단한 프로그램들은 쉽게 만들 수 있지만 고 난이도의 프로그램 구성은 상당한 지식을 필요로 한다. 파이어베이스는 2011년 파이어베이스사가 개발하고 2014년 구글에 인수된 모바일 및 웹 애플리케이션 개발 플랫폼이다[9]. 파이어베이스는 실시간으로 클라우드 데이터와 동기화를 이루도록 Backend 서비스를 지원해준다. 만약 어떤 클라이언트가 클라우드 데이터를 CREATE, UPDATE, DELETE 하였다면, 해당 데이터 스코프 영역을 fetch 하는 다수의 클라이언트에게 자동으로 동기화 명령을 내리도록 하여 모든 클라이언트간의 데이터가 일치 되도록 해준다. 즉 데이터베이스를 온라인 공유 변수로 하여 개발할 수 있다. 파이어베이스 호스팅은 앱에 운영할 정적인 파일(html, css, js, media등)을 cdn 형식으로 리소스를 배포할 수 있도록 지원한다. php, JSP, node.js 와 같은 동적인 애플리케이션을 운영할 순 없지만, 파이어베이스 데이터베이스를 통해 Client Side 코드(javascript) 로서 동적인 결과물을 생성할 수 있으므로 Dynamic Web Page 제작도 가능하다. 파이어베이스 심플 로그인은 페이스북, 트위터 등 소셜 로그인을 지원하여 간단하게 유저 인증을 거칠 수 있도록 지원해주며 jwt(json web token)를 통해 custom 로그인을 수행할 수도 있다. 파이어베이스 로그인을 통해 파이어베이스 서비스를 유저의 지정된 범위 안에서 read, write 권한을 부여하여 운영할 수 있다[10].

III. Simulation App Design

인간의 기억은 보고-듣기만 할 때 보다, 보고-듣고-실습할 때 더 많은 내용을 기억한다[11]. 따라서 본 앱은 시각적 효과와 상호 작용적 효과를 얻기 위해 만들었다.

1. Design Direction

본 연구자의 관점에서 학습자들에게 흥미와 이해의 증진, 사고력의 향상에 도움을 주기 위하여 갖추어야할 조건으로 판단한 시뮬레이션 환경은 다음과 같은 점들이다.

시뮬레이션 환경 조건을 정확히 구성해야 한다. 첫째, 하나의 다중 프로그래밍 운영체제 통제 하에서 공유 변수에 접근하는 다수의 프로세스들 형태를 구성해야 한다. 앱이 수행되는 스마트폰은 하나의 프로세스가 되고, 공유변수는 모든 프로세스가 접근하여 읽거나 쓸 수 있는 클라우드 상에 공유 변수를 구현한다. 앱 별로 공유변수를 생성할 수 있게 하고, 변경할 수 있게 하며, 삭제할 수 있게 한다. 둘째, 하나의 운영체제하에서 프로세스들이 통제를 받아서 수행하는 모습을 구현해야 하는데 이를 위해 랩(lab) 개설자와 참가자로 구분하여 참여하도록 하고, 랩 개설자가 운영체제의 역할을 하는 것으로 한다. 셋째, 각각의 프로세스들이 각각의 명령들을 어떤 차례로 수행하더라도 결정성을 유지하듯이 각각의 앱을 하나의 프로세스로 할 때 임계 영역 처리 수행되는 환경을 구성한다. 넷째, 다수의 학습자가 모둠을 구성하고 각각의 모둠이 실습 환경이 되게 하여 각 환경에서 실습할 수 있게 한다.

임계영역 문제는 다음과 같이 '잔류영역1, 진입영역, 임계영역, 진출영역, 잔류영역2'의 5개 영역으로 구성하여 해결한다. 구성하는 시뮬레이터 환경도 이 구성이 반영되어야 하며, 각 프로세스의 실행중인 코드가 어떤 영역에 있는지 보여주어야 한다. 또 각 영역에서 명령을 수행하여 Test-and-Set 알고리즘 이해에 의미 있는 변수에 값의 변화가 일어나면 값의 변화를 모든 프로세스가 인지할 수 있어야 한다. 이는 자신 프로세스의 명령어 실행 뿐 아니라 다른 프로세스의 명령어 실행도 반영되어야 한다. 다른 프로세스의 명령어 수행은 실시간 동적 처리의 반영이 되어야 한다.

마지막으로 본 연구에서 구성되는 시뮬레이터 앱은 학습자의 의도를 반영하는 시뮬레이터가 되어야 한다. 잔류 영역1에 있다가 임계 영역으로 들어가고자 할 때 먼저 진입 영역을 거쳐야 하므로 진입 영역 들어가기 의도를 반영할 수 있어야 하고, 진입 영역에서 임계 영역으로 들어가기는 프로세스간 활동에 의하여 구성한 코드의 수행에 의하여 자율적으로 들어가게 하나, 임계영역에서 진출 영역으로 나오는 것은 학습자의 의도를 반영하게 해야 하고, 진출 영역에서 나오는

것은 진입 영역에서의 활동과 같이 구성한 코드의 자율적 수행에 의하여 나오게 할 수 있어야 한다. 이런 상황에서 다시 임계 영역으로 들어가고자 한다면 다시 임계 영역1에 있는 것과 같은 상황이 되어 반복 될 수 있어야 한다.

2. Design

컴퓨터상의 자원을 공유하면서 병행 수행하는 다중 프로그래밍 환경에서도 모든 프로세스는 바르게 작동되어야 한다. 다른 말로 표현하면 각 프로세스에 가한 동일한 입력에 대해 프로세스의 다양한 수행 순서에도 컴퓨터의 처리 결과는 같아야 한다. 이를 결정성이라고 한다. 다중 프로그래밍 환경에서 다수의 프로세스들이 자원을 공유하면서 병행적으로 실행될 경우에 결정성 보장을 위해 특별한 기능이 요구되었다. 이를 임계영역 문제라고 한다. 결정성 보장을 위해 공유 자원을 접근하는 코드 영역을 임계 영역이라고 하며, '임의의 프로세스가 임계 영역 실행을 시작하면 임계영역 실행을 종료할 때까지 방해 받지 않고 수행하도록 하는 보장'이 모든 프로세스에 적용되어야 한다는 것이다. 이를 해결하기 위해 많은 노력을 하였고 많은 결과를 만들었으며, 노력의 결과가 모두 완벽한 것은 아니다. 문제점의 범위를 작게 인지하였을 경우 그 범위만 해결하는 알고리즘이 만들어 졌는데, 또 다른 문제점들이 발견된 것이다. 이렇게 노력하는 동안에 임계 영역 문제 해결 3 조건이 찾아졌으며 그 해결 조건은 다음과 같다.

조건1) 상호 배제(mutual exclusion) : 임계 영역 안에 프로세스가 있으면 다른 프로세스의 진입은 거부 되어야 한다. 조건 2) 진행(progress) : 임계영역 안에 프로세스가 없으면, 진입하려는 프로세스는 진입할 수 있어야 한다. 조건 3) 제한된 대기(bounded waiting) : 진입을 원하는 프로세스에게 무한정 기다리게 해서는 안 된다[12-13].

Table 1은 임계영역 처리를 위하여 개발된 알고리즘들에 대한 임계 영역 문제 해결 3조건에 대한 해결 여부를 표시한 것이다.

Table 1. System Environment

알고리즘	프로세스 수	Mu	Pr	Bo	비고
Dekker Algorithm 1	2	Sa	Dis	Dis	s/w method
Dekker Algorithm 2	2	Sa	Dis	Dis	s/w method
Peterson Algorithm	2	Sa	Sa	Sa	s/w method
Bakery Algorithm	1 or more	Sa	Sa	Sa	s/w method
Test-and-Set	1 or more	Sa	Sa	Sa	h/w method
Swap	1 or more	Sa	Sa	Sa	h/w method
Semaphore	1 or more	Sa	Sa	Dis	Depends on P V operation

*. Mu : Mutual exclusion, Pr : Progress, Bo : bounded waiting, Sa : Satisfaction, Dis : Dissatisfaction

잔류 영역 1	임계 영역 들어가기 전 수행 코드

진입 영역	waiting[i]=TRUE;
	key=TRUE;
	while (waiting[i] && key)
	  key=Test_and_Set(&flag);
	waiting[i] = FALSE;

임계 영역	임계 영역에서 수행해야할 코드

진출 영역	j = (i+1) % N;
	while((j != i) && (waiting[j] == FALSE))
	  j = (j+1)%N;
	if (j == i)
	  flag = FALSE;
	else
	  waiting[j] = FALSE;

잔류 영역 2	임계 영역에서 나온 후 수행할 코드

Test-and-Set	bool Test_and_Set(bool * lock)
	{
	  bool ret;
	  ret = *lock;
	  *lock = TRUE;
	  return ret;
	}

*. key, j:지역 변수, i:프로세스 번호, N:최대 프로세스 수, flag:공유변수, waiting:공유변수로서 크기가 N인 배열

Fig. 2. Test-and-Set Algorithm Code

Test-and-Set 알고리즘은 아래와 같이 알려져 있다. 임계 영역에 들어가기 위해서는 반드시 진입 영역을 거쳐야 하며, 또한 임계 영역에서 나오면서 반드시 진출 영역을 거쳐야 한다. Test-and-Set 알고리즘은 기계어 명령이라는 조건하에 수행되는 임계 영역 처리 방법으로 Test-and-Set 명령이 수행되는 동안은 기계어라는 조건 때문에 인터럽트도 되지 않고 문맥 교환도 일어나지 않는다. 이 알고리즘은 컴퓨터 내부에서 수행되는 운영체제의 일부로 수행 행동을 관찰하기 어려운 수행 과정이다. 또한 이 알고리즘이 채용된 운영체제에서 자동으로 수행되는 것으로 이 상태에서는 의도를 반영하기도 어렵다. 이러한 어려움을 보완하면서 교육적인 효과를 거두기 위해 수행 과정에 의도를 반영할 수 있게 앱을 설계하였다. 그 설계 목표는 다음과 같다. 첫째 각 프로세스의 필요한 단계 마다 진입 의도를 반영할 수 있으며, 둘째 알고리즘 동작 시 필요한 공유 변수 및 지역 변수의 변화를 볼 수 있게 한다. 또한 한 번의 수행으로 이해가 되지 않을 때 동일 한 상황을 만들어 몇 번이라도 다시 볼 수 있도록 한다. 셋째, 학습자가 많음에도 실습에 문제없게, 동시에 여러 팀이 각각의 실습 환경을 구성하여 실습을 할 수 있게 한다. Fig. 3.은 설계 중 요구조건을 분석하여 그린 Usecase Diagram 이다.

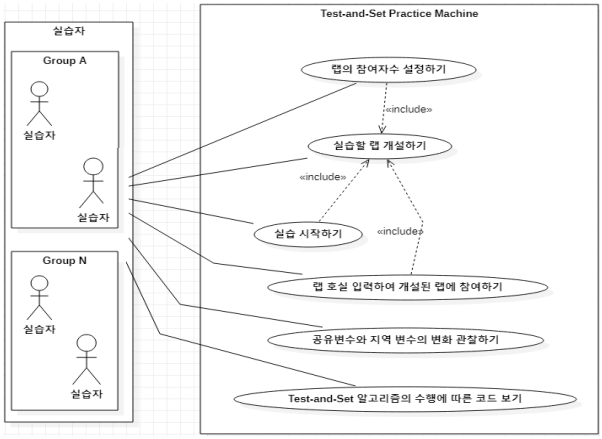

Fig. 3. Usecase Diagram

요구조건에서 보인 바와 같이 동시에 여러 팀이 각각의 Test-and-Set 실습 환경을 구성할 수 있게 하였다. 본 앱의 작동 환경은 인터넷상의 데이터베이스와 본 연구에서 구성한 앱을 설치한 스마트폰들이다. 스마트폰의 앱 각각은 실습 환경에서 프로세스로 작동한다. 여러 개의 스마트폰 앱이 모여 Test-and-Set 알고리즘에 의한 하나의 실습 환경을 랩으로 구성하되, 각 팀의 구별은 데이터베이스에 기록되는 정보로 처리한다. 개설자로서 랩을 개설할 때 4자리의 랩 호실 번호를 받게 되는데, 이 랩 번호가 데이터베이스에 기록되는 모든 정보에 접두어로 기록되게 된다. 그래서 랩별 정보들이 구별 처리되며, 이 랩에서 작업하기 원하는 모든 앱들도 자신들의 명칭뿐 아니라 모든 자료에 호실 번호를 접두 처리함으로서 여러 팀이 시뮬레이션을 수행하더라도 문제없도록 하였다.

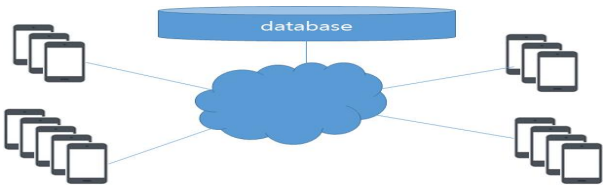

Fig. 4. System Architecture

본 연구에서 개발한 시뮬레이션 환경의 전반적 구조는 Fig. 4와 Fig. 5와 같다.

개설자 또는 참가자로서 수행 과정은 Fig. 6.과 같으며, Test-and-Set 알고리즘은 Fig. 2.와 같으므로 흐름도를 생략하였다.

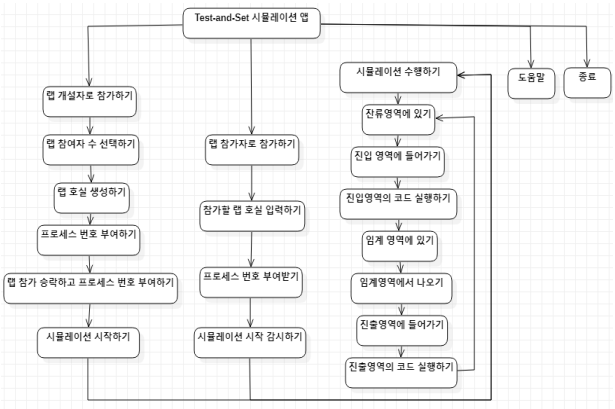

Fig. 5. Design Diagram

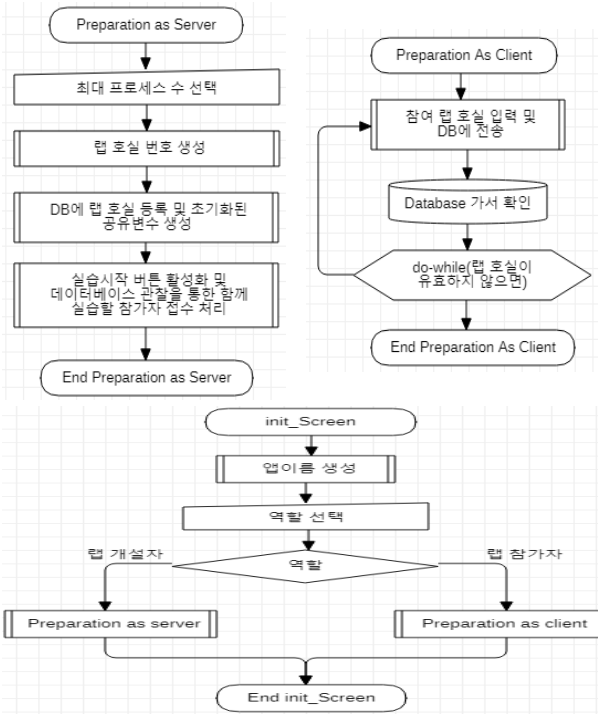

Fig. 6. Flowchart

IV. Experiments and Evaluation

Fig. 7부터는 구현한 앱이 실행되는 과정으로 동시에 여러 랩들이 실습할 수 있게 한 모습을 볼 수 있다. 랩 개설자와 참가자 버튼을 통해 개설자와 참가자가 구별되며, 폰이 앱 단위이므로 한 학습자가 여려 폰으로 한 랩을 구성하여 실습할 수도 있다.

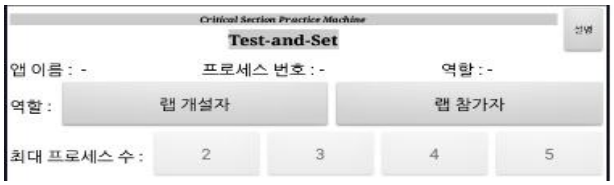

Fig. 7. Running Screen

개설자 버튼과 최대 프로세스 수를 누르면 랩의 호실과 Test-and-Set 알고리즘을 수행하기 위한 관리용 변수들과 공유 변수들이 서버에 생성된다. Fig. 8.는 데이터베이스 상에 2개의 랩이 형성된 모습으로 총 4명의 실습 참가자 중 2명이 개설자로서 2개의 랩을 구성하였다.

Fig. 8. Shared Variable and Management Variable: 2 Lab Configuration

최대 2개의 프로세스까지 허용하는 3903 랩과 최대 4개의 프로세스까지 허용하는 2778 랩을 구성하였고, 다른 2명은 C299, G727의 이름으로 4개까지의 프로세스를 허용하는 랩에 참가하였고 3번과 2번으로 프로세스 번호를 부여 받은 상황이다. 또한 각 랩별로 RegisteredProcess를 통해 등록된 프로세스들을 볼 수 있고 공유변수 flag와 waiting 변수가 형성되어 있고, Test-and-Set의 수행 결과가 저장될 key 변수들이 구성되어 있음을 볼 수 있다. maxNumOfProcess는 그 랩의 최대 허용치로, 그 이상이 들어오려고 할 때 통제할 수 있는 변수이다. 실습시작 태그는 참여한 프로세스들에게 시뮬레이션 시작을 알리는 방송용으로 준비되었다. 여기에서 방송이란 임의의 프로세스가 데이터베이스에 기술하는 것을 다른 프로세스들이 읽어가는 것을 의미한다.

Fig. 9. (1)의 3개 앱 화면에서 앱 이름과 프로세스번호, 랩에서의 역할 기록을 데이터베이스 자료와 비교해 볼 수 있다. 개설자는 프로세스 번호를 1로 부여 받았고, 개설된 랩 호실은 2778, 최대 프로세스는 4로 설정하였는데 현재 자신을 포함한 오른쪽의 그림과 같이 두 개의 프로세스가 더 들어왔으므로 추가로 더 받을 수 있는 여유는 1로 표현되어 있음을 확인할 수 있다. 오른쪽 참가자 프로세스들도 같은 방법으로 표현되어 있음을 확인할 수 있다. 아울러 개설자만 실습을 시작할 수 있게 실습 시작 버튼이 활성화되어 있다. 랩을 개설을 한 후 개설자 프로세스는 데이터베이스 값 변화를 확인하여 입실을 원하는 참가자를 살펴보며, 참가자 프로세스들은 '자기가 참가하기 원하는 랩에 입장이 허락되었는지.'와 허락된 후 자신이 속한 랩의 실습이 시작되는지를 살펴본다. 이 행동들은 실습 시작 버튼이 눌릴 때까지 계속된다. 개설자의 최대 프로세스는 의미 그대로 최대 프로세스이며, 언제든지 실습 시작 버튼을 눌러 Test-and-Set 알고리즘을 시작할 수 있다. 개설자만이 실습 시작을 할 수 있으며, 개설자가 실습 시작을 누르면 데이터베이스의 '실습시작' 태그가 true로 되며, 데이터베이스 값 변화를 살펴보던 참가자 프로세스들은 실습 시작 태그 값의 변화를 인지하고 실습을 할 수 있는 상태가 된다. Fig. 9. (2)는 개설자가 '실습시작' 버튼을 눌러서 참가자를 포함한 모두가 자동으로 실습을 시작할 수 있는 상태가 된 것이다. 앱 화면에 공유 변수로서 flag와 waiting 배열이 보이며, flag는 초기 값으로 false로 설정되어 있고, 최대 프로세스를 4으로 하였으므로 waiting은 4개까지만 false로 설정된 상태이다. 지역 변수로는 N은 최대 프로세스 수가 4임의 표시하며, i는 각각의 프로세스 번호를 표현하고, key 변수와 j 변수는 현재는 무관 상태라 값이 의미가 없음의 표현으로 '-'로 표현된 것을 볼 수 있다. 각 프로세스들의 위치를 '잔류 영역 / 진입 영역 / 임계 영역 / 진출 영역'으로 구분하고 있는데 '잔류 영역 1'과 '잔류 영역 2'는 구분할 필요가 없어 하나로 표현 하였으며, 따라서 진출 영역을 나오면 다음 수행 코드는 첫 위치에 표현한 잔류 영역이다. 좌측의 '진입영역으로 들어가기 버튼'과 '임계영역에서 나오기' 버튼은 Test-and-Set 알고리즘 수행에서 수행의 관찰을 위해 실습자의 의도를 반영하기 위하여 준비하였다. 잔류영역에서 임계 영역으로 들어가고 싶을 때 진입 영역을 거쳐 임계 영역으로 들어가야 하므로 잔류영역에 있는 상태에서 실습자가 표현할 수 있는 의도는 진입 영역에 들어가기이며,

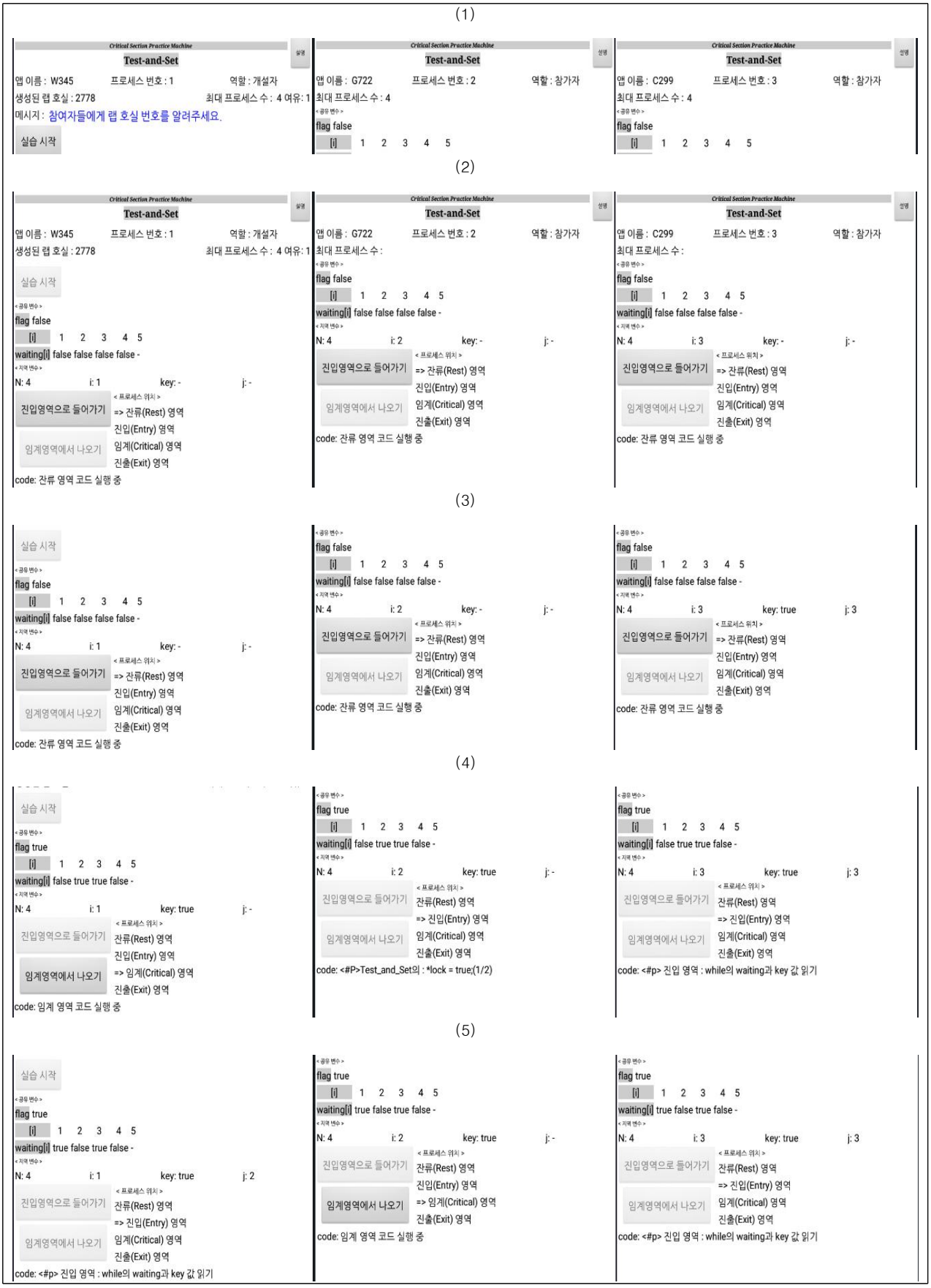

Fig. 9. Running Screen

따라서 이 상황에서는 '진입영역으로 들어가기' 버튼이 활성화되도록 하였다. 현재 각각의 수행 상태는 모두 잔류영역에 있음을 표현하며 따라서 현재 실행중인 코드도 잔류영역 코드를 실행하는 중으로 표현되어 있다. Fig. 9. (3)은 프로세스 3이 임계 영역을 들어갔다가 잔류 영역으로 다시 온 상태이다. 그림에서 key 값과 j 값이 변화된 모습을 볼 수 있다. j의 값이 3인 이유는 프로세스 3은 임계 영역에서 나오면서 진출 영역에서 모든 프로세스의 임계영역 진입 의도를 살펴본다. 각 프로세스는 앞에서 보여준 코드와 같이 자신의 임계 영역 진입 의도를 waiting 배열에 표현한다. 따라서 프로세스 3은 데이터베이스의 waiting 배열을 살펴보았고, 임계 영역 진입 의도를 표현한 것이 없어 자신의 프로세스 번호와 같은 3이 되었다. 진입 의도 파악은 앞에서 표현한 코드와 같이 j 변수를 이용하여 자신의 프로세스 번호 다음부터 차례로 확인하며 모듈러 연산을 통해 자기 자신까지 오면 확인을 끝낸다. 따라서 j의 값이 프로세스 번호와 같이 3이라는 이야기는 임계영역에 진입하려는 프로세스가 없었다는 것이다. 따라서 프로세스 3은 다시 임계 영역에 진입을 할 수 있으며, 본 연구에서 개발한 실습 장치가 임계 영역 문제 3조건 중 2번째 조건인 '진행 (progress) 조건'이 처리되고 있음을 알 수 있다. Fig. 9. (4)는 앞 그림 상태에서 모든 프로세스가 진입영역으로 들어가기를 버튼을 클릭한 상태에서 프로세스 1이 임계 영역으로 들어갔고, 다른 두 프로세스는 진입 영역에서 임계 영역에 들어가기 위한 수행을 하고 있는 중임을 보이고 있다. 따라서 프로세스1만 임계영역에서 나오기 버튼이 활성화되었고, 나머지는 비활성화 되어 있다. 변수의 변화는 공유 변수 flag 값이 true로 변형되어 임계 영역에 누군가가 있음을 표현하고 있고, 프로세스 1이 임계 영역 안에 있어 첫 번째 waiting 값이 false이나 2번과 3번 프로세스는 임계 영역에 들어가기를 희망하고 있어 해당 waiting 값들이 true로 설정되어 있음을 볼 수 있다. 3명만 실습에 참여하여, 4번째 waiting 실습에 참여하지 않은 상태이고 임계 영역 안에 들어가기를 희망 할 수 없으니 늘 false로 설정되어 있다. 2번 프로세스의 현재 수행 중인 문장은 Test-and-Set의 '*lock = true;' 명령어를 수행하고 있고, 프로세스 3은 진입 영역의 while의 조건식 안에 waiting과 key 값을 읽고 있음을 보이고 있다. 위 실행이 데이터베이스에서도 확인 할 수 있다. Fig. 9. (4) 실행의 의미는 임계 영역 안에 어떤 프로세스가 있으면 다른 프로세스는 진입이 거부되어야 한다는 상호배재(mutual exclusion) 조건이 구현되었음을 알 수 있다. Fig. 2.에 함수와 같이 기술된 Test-and-Set은 기계어 명령으로 중단되지 않고 처리 된다고 가정하고 있다. 이를 처리하기 queue 리스트를 준비하였고, 시작 전에 자신의 프로세스 번호를 여기에 입력하고, queue 리스트에 기록되어 있는 한 인터럽트 되지 않게 하였다. Fig. 10.에서 데이터베이스의 Test-and-Set이 작동 중임을 색의 변화로 알 수 있고, queue 태그를 통해 2번 프로세스가 Test-and-Set을 실행하고 있음을 볼 수 있다.

Fig. 10. Shared Variables and Management Variables: Test-and-Set in Execution

Fig. 9. (5)는 Fig. 9. (4)의 상태에서 프로세스1이 임계영역에서 나오면서 대기 하고 있던 프로세스2의 waiting 값을 false로 만들어 주어 프로세스2가 임계영역으로 들어간 상태를 보여주고 있다. 프로세스1은 좀 더 진행을 하여 다시 진입 영역에 들어가 임계 영역으로 들어가기 위해 수행 중이다. 프로세스2와 프로세스3이 모두 임계 영역에 들어가고자 하였으나 수행중인 프로세스1이 자신의 프로세스 다음 번호부터 대기하는 프로세스를 wake up(waiting 값을 false로 만들어 주기)을 하므로 프로세스3을 제치고 다음 번호인 프로세스2가 임계 영역에 들어간 것이다. 알고리즘은 큰 프로세스 번호 순으로 탐색을 하다가 끝에 오면 modulo 연산으로 처음으로 와서 다시 큰 프로세스 번호 순으로 탐색을 하므로 진입을 원하는 프로세스는 무한정 기다림이 없이 최대 '프로세스 수 - 1' 정도의 수행 후에는 반드시 임계 영역으로 들어갈 수 있다. 따라서 본 구현이 '제한된 대기(bounded waiting)'가 구현되었음을 알 수 있다.

V. Conclusions

본 연구에서 개발한 시뮬레이션 앱은 임계 영역문제를 해결하기 위하여 만든 Test-and-Set 알고리즘이므로 시뮬레이션 시 임계 영역문제가 요구하는 3가지 조건이 모두 잘 수행되어야 한다. 4장에서 이것들이 잘 수행되고 있음을 보

였을 뿐 아니라, 의도를 반영하여 수행하고, 수행 과정중인 명령어와 각 변수들의 변하는 모습을 보여 수행하는 모습을 보여주었으므로 학습자들의 이해에 도움이 되길 바란다.

향 후 현장에서 학습자들이 사용하면서 요구하는 내용을 반영한 개선된 유저인터페이스를 추가한 상태로의 개선이 필요하다. 아울러 이를 바탕으로 시뮬레이션 앱 뿐 아니라 다양한 종류의 교육용 앱들이 개발되어 학습자들에 큰 도움이 되길 바란다.

REFERENCES

[1] JinGu Jung, Consumer Post, http://www.consumerpost.co.kr/news/articleView.html?idxno=207946

[2] Hoang Youn-ju, “A Study on the 3D Augmented Reality Animation Education Method for Elementary School Arts Using Smartphone Application ‘Quiver’”, Korean Elementary Art Education Association, Vol. 60, No. 0, pp. 277-308, 2020 02.

[3] Ahn Sung-ho, “Research on Teaching Effective Smart-phone Video Production in Film Production Classrooms”, ASIAN CINEMA STUDIES, Vol. 10, No. 2, pp. 161-179, 2018 04.

[4] Eun Joo Yang, Min Sun Kang, “Development of Music-Science Convergence Program(STEAM): Focusing on Popular Music Instrument Making and Smart phone Application Composition”, Korean Association of Arts Education, Vol. 13, No. 3, pp. 206-220, 2015 09.

[5] Kim, Seok-Hun; Woo, Hee-Sun, “Design of Children and Adolescent's Parents for oral health convergence education App in Mobile Environments”, Journal of The Korea Society of Computer and Information, Vol. 22 No. 1, pp. 57-62, Janurary 2017.

[6] Su-jeong Jeong, Keol Lim, Yujung Ko, Hyunae Sim, Kyungyeon Kim, “The Analysis of Trends in Smart Phone Applications for Education and Suggestions for Improved Educational Use”, Journal of Digital Contents Society, Vol. 11, No. 2, pp. 203-216, Jun. 2010

[7] Yeonjung Joe, Seongyoun Hong, “Education Culture Research”, Inha University Education Research Institute, Vol. 20, No. 3, pp. 99-124, 2014

[8] WIKIPEDIA, App Inventor for Android,https://ko.wikipedia.org/wiki/%EC%95%B1_%EC%9D%B8%EB%B2%A4%ED%84%B0

[9] WIKIPEDIA, Firebase, https://ko.wikipedia.org/wiki/%ED%8C%8C%EC%9D%B4%E C%96%B4%EB%B2%A0%EC%9D%B4%EC%8A%A4

[10] S.team, Firebase, https://s-team.tistory.com/3

[11] Choi, Yun-Chul, “Multimedia Course”, SengNeungChulPan, pp.28-29, 2014

[12] Im, SungRag; Jung, Won-il, “Operating System”, 21SeGiSa, pp.75-98, 2013

[13] Gu, Hyun-He, “Operating System”, HanBit Academy, pp.155-168, 2016

Authors

Kyong-ho Lee received the B.S. degree in Computer Science from Korea National Open University and the M.S. degree in Information and Communication Engineering from Korea Advanced Institute of Science and Technology and the Ph.D. degrees in Electronic Engineering from Dankook University, Korea, in 1991, 1994 and 2008, respectively. Dr. Lee is currently a Professor in the School of Information & Communication, Broadcasting Engineering, Halla University. He is interested in HCI.