Journal of The Korea Society of Computer and Information
Vol. 20, No. 1, January 2015
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2015.20.1.185

웹소켓을 이용한 병원 접수시스템 개선에 관한 연구

손만근*, 박기성*, 공용해*

An Improvement of Hospital Reception System using Web Socket

Man-Geun Son*, Ki-Seong Park*, Yong-Hae Kong*

요 약

병원에서 특정 시각 대에 집중하는 외래환자 접수상황을 효과적으로 해당 진료과에 전달하는 기술적 방식이 기존의 병원 접수시스템에 부재하다. 따라서 모든 진료과는 접수상황을 알기위해서 병원접수 데이터베이스 전체를 반복적으로 순차탐색하게 되고 이는 병원정보시스템 성능저하의 큰 요인이 된다. 따라서 본 논문에서는 이러한 병원 접수시스템의 효율 향상을 위해 웹소켓을 이용한 주키전송 일괄처리시스템과 접수정보전송에 의한 실시간시스템을 개발하였다. 주키전송 일괄처리시스템은 기존의 순차탐색시스템에 비해 데이터베이스 조회속도를 크게 단축하였을 뿐만 아니라 접수환자 수에 무관한 빠른 조회속도를 유지하였다. 또한, 접수정보전송 실시간시스템은 데이터베이스를 조회하지 않고도 기존의 요청/응답 접수시스템의 대기환자 갱신시간을 효과적으로 줄일 수 있었다.

▸ Keywords : 병원 접수시스템, 웹소켓, 데이터베이스 탐색

Abstract

During hospital peak times of outpatients, an effective mechanism that informs the newly receipted patient information to corresponding medical departments is lacking in current hospital reception systems. Since every department repeatedly searches entire patient reception database in sequential manner to acquire its reception information, this is a significant performance degradation factor in hospital information system. To improve hospital reception system efficiency, we developed two web socket based systems, a primary key transmitting batch system and a reception information transmitting real time system. The former reduced database access time compared to sequential search system as well as kept search time low regardless of receipted patient number. The latter effectively reduced waiting list updating

•제1저자 : 손만근 •교신저자 : 공용해
•투고일 : 2014. 8. 28, 심사일 : 2014. 11. 18, 게재확정일 : 2015. 1. 21.
* 순천향대학교 의료IT공학과(Dept. of Medical IT Engineering, Soonchunhyang University)
※ 이 논문은 2010학년도 순천향대학교 교수 연구년제에 의하여 연구하였음

time in request/response patient reception system by eliminating database access.

▸ Keywords : Hospital Reception System, Web Socket, Database Search

I. 서 론

병원정보시스템은 병원의 전반적인 관리 업무를 전산화한 시스템으로 환자의 외래 입·퇴원 관리, 의료·수가 관리, 검사와 같이 병원 종사자들에게 필요한 업무를 전산화한 정보처리시스템이다[1]. 병원정보시스템 중 외래접수시스템은 병원에 내원한 외래환자들을 해당 진료과와 의사에게 접수해주는 업무를 담당하며, 내원한 접수환자가 진료받기 위해 대기하는 시간을 줄이기 위한 진료안내 연구사례를 볼 수 있다[2]. 병원 외래접수 업무는 주로 오전 9시~12시와 오후 1시부터 4시까지 특정 시각대에 집중적으로 발생하므로 병원업무에 병목현상이 생김과 동시에 환자 대기시간 또한 증가하게 되므로 외래접수시스템은 매우 효율적으로 구현되어야 한다. 최근에 웹기반으로 개발되고 있는 정보시스템들은 별도의 클라이언트 설치가 필요하지 않기 때문에 접근권한만 있다면 웹브라우저를 갖춘 어떤 장소에서든 사용이 가능하다. 따라서 병원 접수시스템도 웹기반으로 개발되어 병원 내부뿐만 아니라 외부에서도 업무를 수행할 수 있게 되어야 한다[3].

병원 외래접수시스템은 진료접수를 효율적으로 관리하기 위한 시스템으로 원무과에서는 이를 통해 외래접수가 이루어지고, 진료과에서는 접수된 내역을 확인하여 원활한 진료가 이루어지도록 관리한다. 그림1-1은 기존 병원 접수시스템의 업무흐름으로서, 원무과에서 환자 접수가 일어나면 웹서버를 통하여 데이터베이스에 접수정보를 저장한다. 각 진료과는 접수테이블을 조회하여 해당 진료과 의사에 대기중인 환자를 확인하게 된다. 진료 대기리스트 정보는 이 때 접수날짜, 시간, 진료과, 대기상태와 같은 조건을 사용해서 데이터베이스를 탐색하게 된다. 그러나 이 방식은 모든 레코드를 순차적으로 탐색해야하기 때문에 접수데이터 수가 방대한 병원 접수시스템에는 적합하지 않다. 반면에 주키(Primary key)인 접수번호만을 사용하여 병원접수 데이터베이스를 조회할 수 있게 된다면 외래접수시스템의 효율을 크게 향상시킬 수 있다.

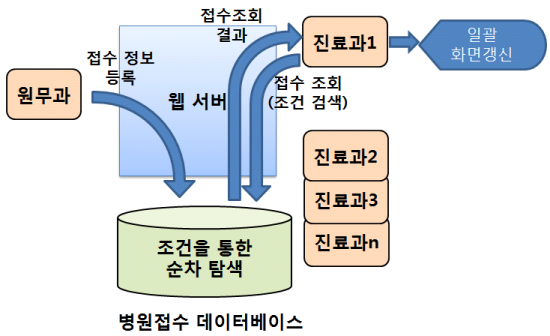

그림 1-1. 순차탐색에 의한 일괄처리 접수시스템
Fig. 1-1. Batch Reception System using Sequential Search

이를 위해서는 각 진료과에서 주키인 접수번호를 실시간으로 수신할 수 있는 기술적 접근이 필요하다. 그림1-2는 주키 전송을 통해 데이터베이스 탐색을 효율화한 일괄처리 접수시스템을 나타낸다. 원무과에서 환자접수가 발생하면 서버는 데이터베이스에 접수정보를 저장하고 해당 접수정보의 주키인 접수번호만을 해당 진료과 클라이언트에 푸시(Push)한다. 진료과 클라이언트는 서버에서 받은 주키정보를 큐에 저장하고 큐에 저장된 주키들을 사용하여 일괄적으로 데이터베이스를 조회하여 대기리스트를 갱신하게 된다.

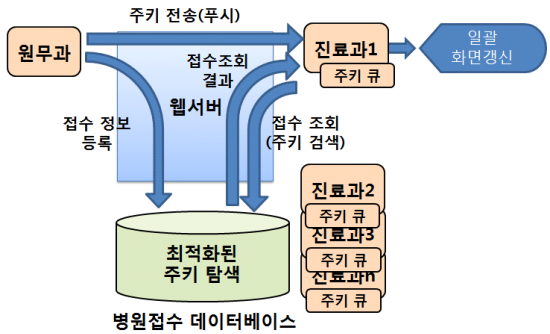

그림 1-2. 주키전송에 의한 일괄처리 접수시스템
Fig. 1-2. Batch Reception System using Primary Key Transmission

웹프로그래밍 환경에서 서버푸시(Server Push)를 구현하기 위해 애이젝스(Ajax)를 사용한 폴링(Polling)이나 롱폴링(Long-polling)을 사용하였다[4]. 그러나 이 방식들은 HTTP 프로토콜의 한계로 인해 많은 트래픽을 발생시키므로 효율면에서 바람직하지 않다[5]. 최근에 웹표준인 HTML5에 정식으로 채택된 양방향 통신 기술인 웹소켓(WebSocket)은 기존 HTTP 프로토콜이 지닌 한계를 극복함과 동시에 트래픽의 감소 효과까지 얻을 수 있으므로 이를 활용하여 병원접수시스템에서 발생하는 조회업무 효율을 향상시키고자 한다.

이를 위하여 jWebSocket 서버를 기반으로 원무과 환자접수페이지와 진료과 외래페이지를 자바스크립트와 HTML5를 사용하여 구현하였고, 접수정보를 데이터베이스에 저장함과 동시에 접수번호를 JSON 형태의 메시지로 웹소켓 서버에서 해당 진료과 클라이언트(외래페이지)로 푸시한다. 진료과 클라이언트는 푸쉬된 접수번호를 자바스크립트 큐에 저장하고, 이를 이용해 접수 데이터베이스를 주키탐색하여 대기리스트 화면을 일괄 갱신하도록 하였다.

본 연구는 더 나아가 각 진료과에서 데이터베이스를 조회하지 않고 진료대기 중인 환자정보를 웹소켓 서버전송 방식으로 접수처에서 실시간으로 전달받을 수 있는 방법을 제안한다. 즉, 그림 1-3과 같이 서버가 새로 접수한 환자의 접수정보를 데이터베이스에 저장하고, 축약된 접수정보를 JSON 메시지로 해당 진료과 클라이언트로 직접 푸시함으로써 데이터베이스 조회 없이 실시간으로 대기리스트 정보를 갱신하게 된다. 상세한 환자 접수정보 조회는 필요시에만 주키탐색을 하게 되므로 불필요한 오버헤드를 크게 줄일 수 있다.

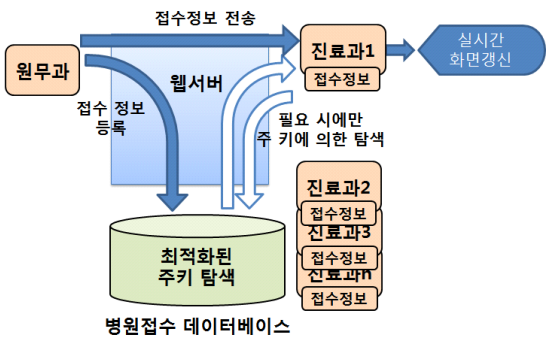

그림 1-3. 접수정보전송에 의한 실시간 접수시스템
Fig. 1-3. Real Time Reception System by Transmitting Reception Information

병원 외래접수 업무는 특정 시각대에 집중적으로 발생하는데 반하여 기존의 병원 접수시스템은 환자접수 상황을 해당 진료과에 효과적으로 알려주지 못하고 있다. 각 진료과가 병원접수 데이터베이스를 수시로 조회하여 환자접수 상태를 파악해야 하는 작업은 병원접수 데이터베이스 전체를 조건에 의한 순차탐색의 반복과정을 수반하므로 방대한 양의 접수정보를 가진 중대형 병원에서의 성능저하 요인이 된다. 본 연구는 접수상태를 효과적으로 조회하는 방법으로 먼저 주키전송에 의한 일괄처리 접수시스템을 제안하고, 병원접수 데이터베이스의 조회성능을 순차탐색 일괄처리 접수시스템과 측정하여 비교한다. 다음으로 접수정보전송에 의한 실시간 접수시스템을 제안하여 데이터베이스를 조회하지 않고도 기존의 요청/응답 접수시스템의 대기환자 갱신시간 문제를 해결하고자 한다.

II. 관련 연구

본 장에서는 병원 접수시스템의 데이터 조회 성능을 높이기 위해서 데이터베이스 조회 시 주키에 의한 탐색 속도와 조건에 의한 순차탐색 속도의 차이를 살펴본다. 또한, 기존의 HTTP 프로토콜을 사용한 통신 방법과 개선된 기술인 웹소켓을 살펴보고 실시간성을 지닌 웹소켓의 장점을 알아본다.

1. 데이터베이스 레코드 조회 방법

데이터베이스를 조회할 때 주키로 탐색하는 것과 그 외의 조건으로 탐색하는가에 따라 검색성능이 크게 차이난다. 주키는 데이터베이스가 레코드를 삽입할 때 물리적으로 최적화된 B-트리나 클러스터드 인덱스(Clustered Index) 등의 탐색구조에 저장되므로 레코드 수에 비례하지 않는 탐색의 최적화가 가능하다. 반면에 조건에 의한 탐색은 데이터베이스를 순차탐색하므로 레코드 수에 비례하여 탐색 시간이 증가하게 된다.

1.1 조건에 의한 레코드 조회 방법

주키가 아닌 일반 칼럼의 탐색은 테이블의 레코드를 순차적으로 탐색하여 레코드를 찾는다. 그림 2-1은 소화기 내과(GI)에 2014-XX-XX일 진료 대기중(W)인 환자를 찾기 위한 과정을 나타낸다.

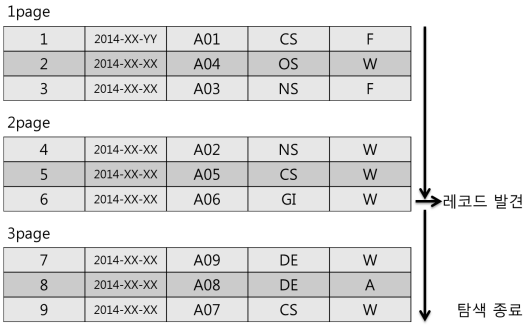

그림 2-1. 일반 칼럼들의 순차탐색
Fig. 2-1. Sequential Search of Field Columns

조건에 필요한 칼럼들이 물리적으로 정렬되어 있지 않으므로 순차적으로 탐색해야할 뿐만 아니라, 탐색조건을 충족하는 레코드의 수를 미리 알 수 없으므로 데이터베이스의 모든 레코드를 끝까지 탐색해야만 한다.

1.2 주키에 의한 레코드 조회 방법

주키를 사용한 칼럼의 탐색은 테이블의 레코드가 삽입 시점에서 물리적으로 구조화되기 때문에 탐색의 속도가 일반 칼럼의 탐색 속도보다 월등히 빠르다.

클러스터드 인덱스 방식으로 구조화된 테이블에서 접수번호가 6인 레코드를 조회하는 예를 그림 2-2에 보였다. 인덱스테이블에서 접수번호 6의 범위를 페이지 2로 가리키므로, 페이지 2에서 해당 레코드를 신속하게 찾을 수 있다. 이와 같이 최적화된 주키탐색은 순차탐색 보다 매우 효율적이다.

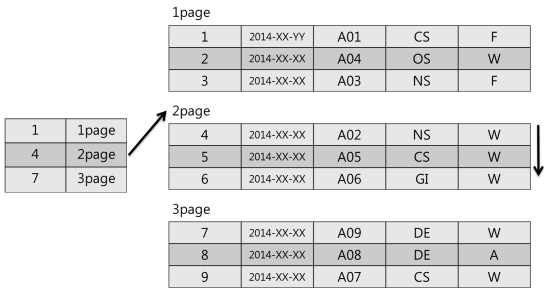

그림 2-2. 주키 칼럼의 최적화된 탐색
Fig. 2-2. Optimal Search of Primary Key Column

2. 기존 HTTP 프로토콜

기존의 HTTP 프로토콜은 HTTP 메시지에 Header(헤더)를 사용하여 서버와 통신하였다. 헤더에는 클라이언트 요청 헤더, 서버 응답 헤더 등이 있으며 상호간에 요청 응답을 통해 데이터를 송수신한다. 모든 이벤트는 클라이언트의 요청이 있을 때에만 발생한다. 서버 측에서는 클라이언트로 직접 푸시할 수 없는 구조이다.

2.1 HTTP 요청/응답(Request/Response) 방식

HTTP 요청/응답 방식은 사용자가 서버 측에 요청을 보내면 서버는 해당 사용자의 요구를 반영하여 웹페이지를 작성하고 이를 다시 응답해 준다. 그림 2-3의 HTTP 요청/응답 과정에서 전송받는 데이터들의 내용은 유사한 경우가 많다. 또한 요청/응답 방식은 클라이언트의 요청이 있을 경우에만 서버에서 응답해준다. 이는 HTTP 프로토콜의 특성상 서버와의 연결이 유지되지 않기 때문으로, 결국 서버 측에서 데이터의 변동이 있더라도 클라이언트는 다시 조회하지 않는 한 변화를 알 수 없다.

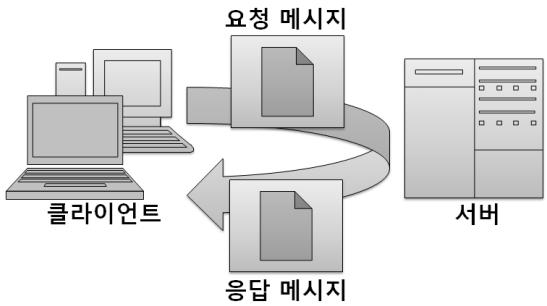

그림 2-3. HTTP 요청/응답 방식
Fig. 2-3. HTTP Request/Response Protocol

2.2 폴링(Polling) 기법

에이잭스(Ajax)를 사용한 폴링 기법은 브라우저에서 클라이언트가 보고 있는 화면과 별개로 자바스크립트(Javascript)를 사용하여 서버와 비동기적으로 그림 2.4와 같이 통신한다. 통신 주기를 일정한 간격으로 설정하여 서버

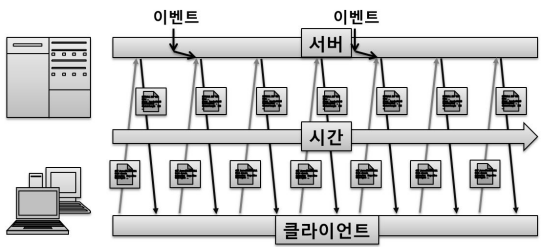

그림 2-4. Polling 기법
Fig. 2-4. Polling Method

에게 반복 요청하여 결과를 얻어낸다. 이러한 방식은 서버의 데이터를 실시간으로 얻어오는 것처럼 보일 수 있지만 서버에 주기적으로 요청을 보내게 되고 결국 불필요한 헤더 전송으로 인해 네트워크에 부하를 일으킨다.

2.3 롱폴링(Long-polling) 기법

롱폴링은 클라이언트가 서버에 최초 연결하면 서버는 이 연결을 응답하지 않고 대기한다. 그 후 특정 이벤트가 발생하면 그림 2.5처럼 클라이언트에게 응답하고 클라이언트는 이벤트를 실행한 후 다시 서버에 연결한다. 이 방식은 서버에서 이벤트 발생 시 응답이 이루어지므로 마치 서버에서 푸시한 것 같은 효과를 내지만 이 방법도 서버에서 이벤트가 빈번하게 발생하는 경우 결국 폴링과 같은 네트워크 부하를 유발한다.

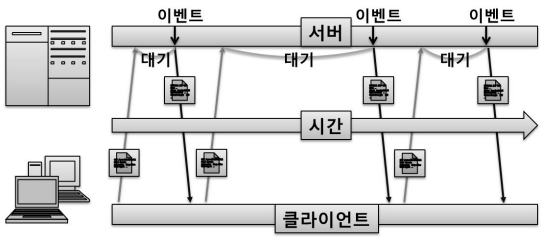

그림 2-5. Long-Polling 기법
Fig. 2-5. Long-Polling Method

3. 웹소켓 프로토콜 (Web Socket Protocol)

이러한 문제를 극복하기 위해 HTML5의 새로운 사양으로 웹소켓(Web Socket)이 제안되었다. 웹소켓은 웹상에서 TCP 연결을 사용하여 전이중 통신이 가능한 기술이다. 또한 HTTP 프로토콜이 아닌 웹소켓 프로토콜을 사용하여 기존의 폴링 기법이나 롱폴링 기법의 불필요한 요청으로 인한 부하를 줄일 수 있다[6].

웹소켓은 브라우저에서 보다 쉽게 양방향 메시지 수신을 하기 위해 등장한 HTML5 표준안의 일부이다. 웹소켓은 HTTP 요청과 마찬가지로 80번 TCP 포트를 사용한다. 최초 연결 시 HTTP 요청의 Upgrade 헤더를 사용하여 웹서버에 요청하고, 서버는 이를 확인하고 토큰을 생성하여 응답한다. 이러한 웹소켓 핸드쉐이킹(Hand Shaking)을 마치면 WS 프로토콜로 변환되어 프로토콜 오버헤드(Protocol Overhead) 방식으로 서버와 브라우저가 그림 2.6과 같이 웹소켓 통신을 하게 된다.

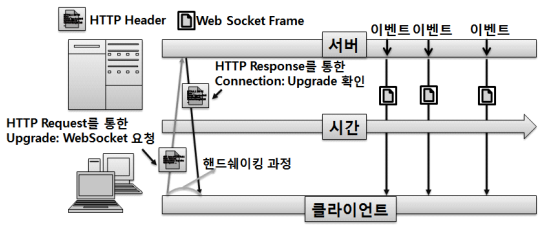

그림 2-6. 웹소켓 기술
Fig. 2-6. Web Socket Technology

웹소켓은 완벽한 전 이중 방식으로 서버에서 클라이언트로 푸시가 가능하다[7]. 또한 기존의 방식들은 매 요청과 응답 시 HTTP 헤더를 주고받아 트래픽의 부하가 심했지만 웹소켓 방식은 최초 핸드쉐이크 과정에서만 요청과 응답 헤더가 있고 그 이후 데이터 송수신에는 최소 2바이트 크기의 프레임을 사용하여 네트워크의 부하도 감소하게 된다[8].

III. 병원 접수시스템 설계

병원 접수업무 분석을 기반으로 데이터베이스를 설계하였으며, 데이터베이스는 접수를 위해 환자테이블과 진료과테이블, 의사테이블, 접수테이블로 구성하였다. 환자접수시스템을 구현하기 위해 서버는 jWebSocket 서버를 사용하였으며, 서버와 클라이언트 간 메시지는 JSON 형태의 메시지로 서버에서는 메시지 구분을 통해 용도에 맞는 이벤트를 실행한다.

1. 접수 및 조회를 위한 데이터베이스 설계

외래 접수정보를 저장하고 관리할 수 있는 데이터베이스의 테이블은 환자 정보 및 의사, 진료과와 같은 기준 정보를 관리할 수 있고 환자들의 외래 접수정보들을 저장할 수 있도록 설계하였다. 그림 3-1은 병원 접수시스템을 위해 설계한 데이터베이스 스키마이다.

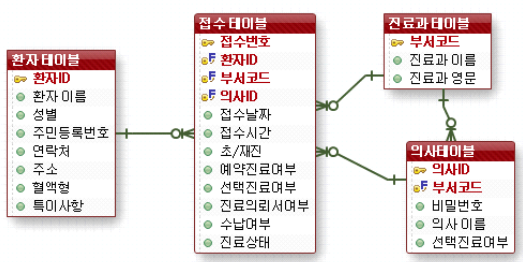

그림 3-1. 환자접수시스템 데이터베이스 스키마
Fig. 3-1. Patient Reception System Database Schema

테이블 구성은 크게 환자 정보를 저장하고 있는 환자 테이블과 의사, 부서 정보를 저장하고 있는 의사 테이블과 진료과 테이블 그리고 접수정보들을 저장하고 있는 접수테이블로 구성하였다. 원무과에서는 환자 이름과 주민등록번호를 통해 환자 테이블에서 환자 정보를 조회하고, 부서코드를 통해 진료과와 해당 의사를 선택하여 외래 접수가 이루어진다. 외래 접수 시 고유의 접수 번호가 주어지기 때문에 접수번호를 통해 접수정보를 확인할 수 있다.

2. 웹소켓 주키전송 일괄처리 접수시스템 흐름

원무과 접수페이지에서는 외래 접수가 이루어지며, 각 진료과 외래페이지에서는 진료 대기 중인 환자들의 대기리스트를 확인한다. 원무과 클라이언트는 자바스크립트를 사용하여 웹 소켓 서버와 연결된다. 웹 소켓은 최초 접속 후 페이지 이동이 없는 이상 연결이 지속된다.

그림 3-2는 웹소켓으로 주키를 전송하는 일괄처리 접수시스템 흐름도이다. 원무과 클라이언트에서 환자를 접수하면 데이터베이스에 접수정보를 저장하고 이 때 생성된 접수번호를 웹소켓을 통해 JSON 형태의 메시지로 해당 진료과 클라이언트에 푸시한다. 진료과 클라이언트와 웹소켓 서버는 실시간으로 연결되어 있으므로 환자 접수와 같은 이벤트 발생 시 메시지를 즉시 수신할 수 있다. 진료과 클라이언트에서는 수신한 접수번호를 자바스크립트 큐에 저장한다. 이를 이용하여 데이터베이스에서 주키탐색으로 환자 대기리스트 정보들을 조회하고, 자바스크립트를 통해 일괄적으로 진료과 클라이언트 화면의 대기리스트를 갱신한다.

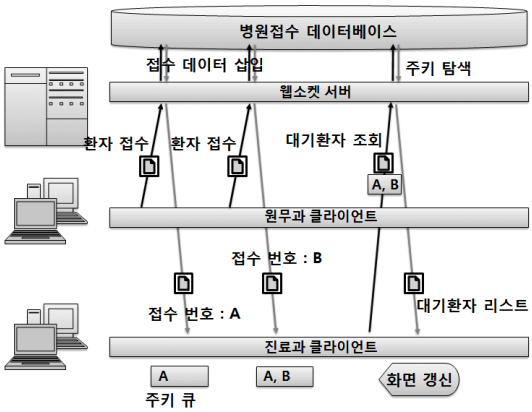

그림 3-2. 주키전송 일괄처리 접수시스템 흐름도
Fig. 3-2. Batch Reception System Flowchart by Transmitting Primary Key

3. 웹소켓 주키전송 일괄처리 접수시스템 설계

그림 3-3은 웹소켓 주키전송 일괄처리 접수시스템의 구조도로서 원무과에서 환자접수 시 접수정보를 병원접수 데이터베이스에 저장하고, 이 때 생성되는 주키인 접수번호를 해당 진료과에 푸시한다. 서버에서 진료과에 접수번호를 푸시를 하기 위해서 서버는 해당 진료과의 클라이언트 주소를 알아야 한다. 이 클라이언트 주소정보는 웹소켓 서버에 진료과 부서코드를 키로 한 주소 해시맵에 저장되어 있다. 키 값으로 쓰이는 부서코드는 원무과에서 외래접수 시 접수된 진료과 부서코드를 접수정보에 입력한다. 서버는 이를 참조하여 진료과 클라이언트의 주소를 확인할 수 있고 환자 접수번호를 해당 진료과 클라이언트에 푸시한다.

진료과 클라이언트는 원무과에서 접수한 데이터들의 접수번호를 큐에 저장한다. 그림 3-3에서는 진료과1 클라이언트에서 접수번호가 1, 5인 환자들을 조회하는 과정을 나타낸다. 진료과에서 환자조회 요청 시 진료과 클라이언트에 저장된 접수번호 큐를 서버에 전달하고, 서버는 이 접수번호를 사용하여 데이터베이스에서 주키탐색을 실행한다. 그림 3-3의 하단에 클러스터드 인덱스 테이블로 예를 보였다. 접수번호 1인 환자정보는 인덱스테이블 참조로 4보다 작으므로 페이지 1을 탐색하여 환자번호 A01인 접수데이터를 찾는다. 접수번호 5는 인덱스테이블에서 4보다 크고 7보다 작으므로 페이지 2에서 환자번호가 A05인 접수데이터를 찾을 수 있다. 이러한 주키탐색은 해당 진료과와 접수일시 등을 조건으로 전체 접수데이터베이스를 탐색하는 것 보다 매우 유리하다. 서버는 주키탐색으로 얻어진 환자 대기리스트 정보를 해당 진료과 클라이언트에게 JSON형태의 메시지로 다시 돌려준다. 진료과 클라이언트는 이 정보를 이용해서 대기리스트 화면을 갱신한다.

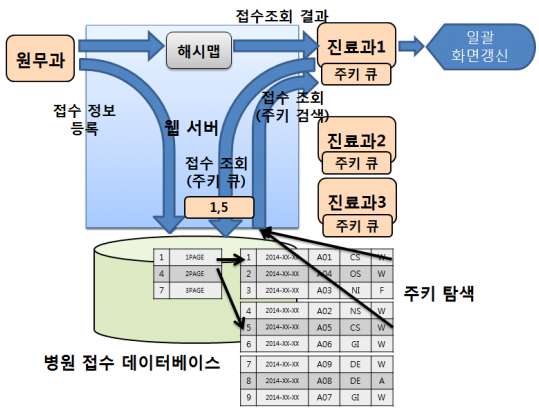

그림 3-3. 주키전송 일괄처리 접수시스템 구조도
Fig. 3-3. Batch Reception System Structure by Transmitting Primary Key

4. 웹소켓 접수정보전송 실시간 접수시스템

그림 3-4는 실시간으로 접수정보를 전송하는 시스템 흐름도이다. 원무과 클라이언트에서 환자를 접수하면 서버는 데이터베이스에 접수정보를 저장하고, 해당 접수정보 중 환자대기리스트 갱신에 필요한 축약된 레코드를 즉시 진료과 클라이언트로 푸시한다. 진료과 클라이언트는 수신한 축약 레코드를 사용하여 환자 대기리스트를 실시간으로 갱신한다.

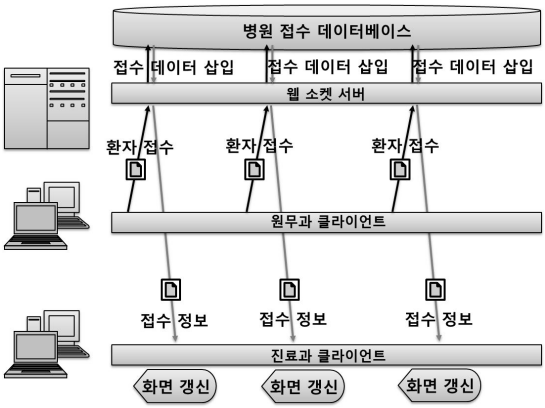

그림 3-4. 접수정보전송 실시간 접수시스템 흐름도
Fig. 3-4. Real Time Reception System Flowchart by Transmitting Reception Information

그림 3-5은 실시간 접수시스템 구조도를 나타낸 것으로 그림 3-3의 일괄처리 접수시스템과 구조적으로 유사하나 다음과 같은 차이점이 있다. 그림 3-3의 시스템은 외래접수 시 서버에서 주키인 접수번호를 진료과 클라이언트에 전달하지

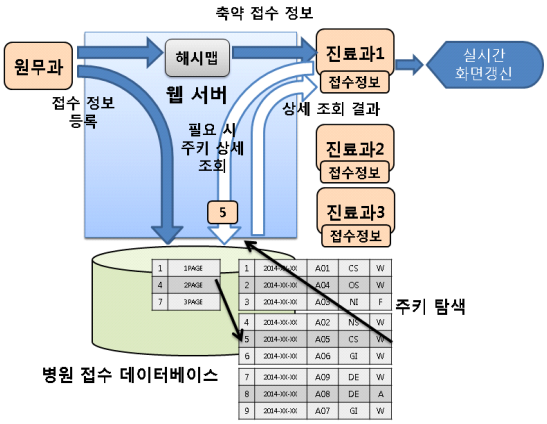

그림 3-5. 접수정보전송 실시간 접수시스템 구조도
Fig. 3-5. Real Time Reception System Structure by Transmitting Reception Information

만, 실시간 환자접수시스템은 접수번호 대신 앞서 설명한 축약된 레코드를 JSON형태의 메시지로 해당 진료과 클라이언트에 전달한다. 진료과 클라이언트는 수신한 접수정보를 실시간으로 환자 대기리스트 목록을 갱신하여 준다. 축약된 레코드에는 주키인 접수번호가 포함되어 있기 때문에 필요시에만 해당 레코드를 조회할 수 있다. 이러한 실시간 접수정보 전송 방법은 진료과 클라이언트가 데이터베이스를 전혀 조회하지 않고도 접수상태를 항상 알 수 있으므로 접수시스템의 효율이 크게 증대된다.

Ⅳ. 접수시스템 구현결과 및 성능측정

병원 접수시스템의 원무과 클라이언트와 진료과 클라이언트는 자바스크립트와 HTML5를 사용하여 구현하였다. 병원 접수시스템의 성능측정을 위하여 데이터베이스에서 환자접수 데이터 조회 시 주키탐색과 조건에 의한 순차탐색에 소요되는 시간을 측정하고 대기리스트 갱신 속도를 비교하여 성능을 평가하였다.

1. 병원 접수시스템 구현결과

1.1 원무과 접수 클라이언트 구현결과

환자정보 조회와 외래접수 기능을 제공하는 원무과 접수 클라이언트는 실시간 데이터 송수신을 위해 웹소켓을 사용하여 서버에 연결한다. 웹소켓의 서버와의 연결은 자바스크립트를 사용하여 간단한 코드로 연결이 가능하고, 이 연결은 별도의 페이지 이동이 없는 한 계속 유지된다.

그림 4-1은 원무과 클라이언트 화면 중 환자 인적사항 조회를 나타낸다. 환자가 원무과에 방문하면 먼저 환자의 이름과 주민등록번호를 통해 환자 테이블에서 환자정보를 조회한다. 그림 4-2는 원무과 클라이언트 화면 중 외래접수를 나타낸다. 진료과 목록에서 진료과를 선택하면, 부서코드를 통해 데이터베이스에서 해당 진료과에 소속된 의사들의 목록을 조회한다. 전문의 목록에서 진료의를 선택하고, 초/재진 여부와 같이 접수에 필요한 내용을 차례로 기입하고 접수 버튼을 누르면 접수 정보는 데이터베이스 접수테이블에 저장되어 외래접수가 완료된다. 외래접수가 완료됨과 동시에 원무과 클라이언트는 접수정보를 표 4-1과 같은 구조의 JSON 메시지로 실시간 연결되어있는 웹소켓을 통해 서버에 전송한다.

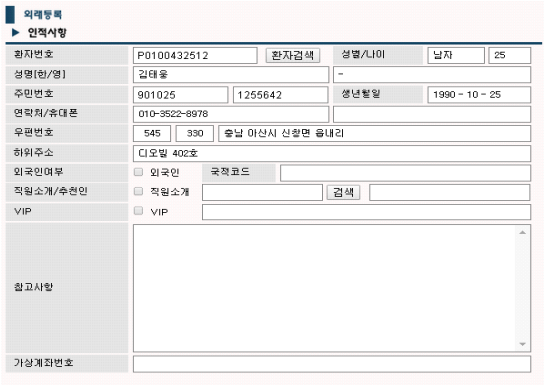

그림 4-1. 환자 인적사항 조회
Fig. 4-1. Patient Information Inquiry

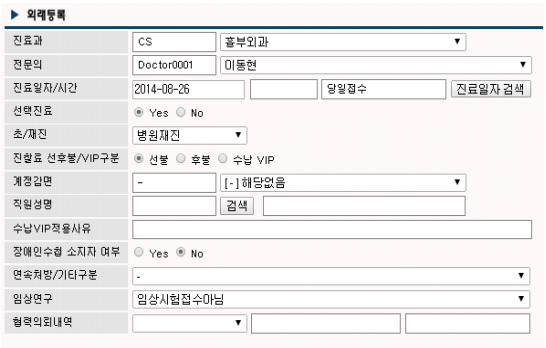

그림 4-2. 외래 접수
Fig. 4-2. Outpatient Reception

표 4-1. JSON 메시지의 구조 예시
Table 4-1. Example of JSON Message Structure

Key	type	receipt_id	depart
Value	메시지 구분	접수번호	부서코드

1.2 진료과 조회 클라이언트 구현결과

그림 4-3은 초기의 비어있는 진료과 클라이언트 화면으로 진료과 클라이언트는 환자대기리스트 조회 버튼을 통해 진료 대기 중인 환자들의 리스트를 조회할 수 있으며, 원무과 클라이언트와 동일한 방법으로 웹소켓 서버와 연결한다.

진료과 클라이언트는 웹소켓 서버와 실시간으로 접속이 유지되어 있기 때문에 서버에서 이벤트 발생 시 즉시 메시지를 수신할 수 있다. 진료과 클라이언트는 서버와 마찬가지로 수신된 메시지의 용도를 확인하고, 외래 접수인 경우 접수번호를 클라이언트 내 자바스크립트 큐에 순차적으로 저장한다.

그림 4-3. 진료과 클라이언트(일괄 조회)
Fig. 4-3. Department Client(Batch Processing)

진료과 클라이언트에서 조회 버튼을 선택하면 큐에 저장되어 있는 접수번호 정보들을 서버에 전송하고, 서버는 이를 이용하여 데이터베이스에서 주키탐색으로 접수테이블에서 환자 대기리스트 내역들을 조회한다. 접수테이블에서 조회한 정보는 표 4-2와 같다.

표 4-2. 접수 테이블 내용
Table 4-2. Reception Table Contents

구 분	내 용	비 고
1	접수 번호	예) A1406359422
2	진료의	예) 홍길동
3	진료 날짜	예) 2014-07-26
4	진료 시간	예) 16:45
5	환자 번호	예) P15212
6	환자 이름	예) 고길동
7	초/재진	예) 초진, 재진
8	수납 여부	예) Y, N

조회한 대기리스트 정보는 서버에서 JSONArray에 저장하여 JSON 메시지로 다시 진료과 클라이언트에 푸시해주고, 진료과 클라이언트는 이를 수신하여 자바스크립트를 사용해 그림4-4와 같이 진료 예정자 리스트에 환자 대기리스트를 갱신해준다.

그림 4-4. 주키탐색에 의한 대기리스트 일괄 갱신
Fig. 4-4. Batch Waiting List Update by Primary Key Search

1.3 실시간 진료과 조회 클라이언트 구현결과

그림 4-5은 실시간 진료과 클라이언트 화면이다. 그림

4-3의 진료과 클라이언트 화면과 동일하게 구현하였으나, 환자대기리스트 조회 버튼을 통해 수동적으로 환자 대기리스트를 갱신하던 이전 방식과 달리 실시간으로 환자 대기리스트를 갱신한다.

원무과에서 외래 접수 시 서버는 해당 진료과 클라이언트에게 JSON형태의 메시지로 축약된 접수정보를 전달하며, 진료과 클라이언트는 수신한 메시지를 자바스크립트를 사용하여 화면에 보여준다. 그림 4-6은 외래접수에 따른 실시간 환자 대기리스트 갱신을 나타내는 것으로 원무과에서 외래 접수가 이루어지면 진료 예정자 리스트에 실시간으로 접수된 환자의 정보가 갱신된다.

그림 4-5. 진료과 클라이언트(실시간 조회)
Fig. 4-5. Department Client(Real Time Processing)

그림 4-6. 실시간 접수정보전송에 의한 대기리스트 갱신
Fig. 4-6. Realtime Waiting List Update by Transmitting Reception Information

2. 병원 접수시스템 성능측정 결과

본 논문에서는 병원 접수시스템에서 접수날짜, 진료과, 대기상태와 같은 조건에 의한 순차탐색과 주키인 접수번호를 사용한 주키탐색을 통해 데이터베이스 환자 대기리스트 갱신 속도를 측정하여 비교하였다.

2.1 성능측정 시나리오

성능측정 시나리오는 다음과 같다.

1. 환자가 원무과에 내원하여 정형외과에 외래접수를 한다.
2. 정형외과 데스크에 있는 간호사는 조회 버튼을 통해 정형외과에 접수된 환자 대기리스트 목록을 조회한다.

3(1). 접수날짜, 진료과, 진료상태를 조건으로 데이터베이스에서 환자 대기리스트 목록을 조회한다.

3(2). 원무과에서 접수 시 서버로부터 전달받은 주키인 접수번호를 사용하여 데이터베이스에서 환자 대기 리스트 목록을 조회한다.

3(1)의 방법과 3(2)의 방법으로 환자 대기리스트 목록의 조회에 소요된 시간을 각각 측정하여 비교한다. 성능비교를 위해 접수테이블에 누적된 접수데이터 수를 10만개부터 10만개 씩 증가시켜 100만개까지 실험하였고, 진료과에 접수된 대기환자 수는 1명부터 20명까지 증가시키면서 환자 수 별 3회 반복 실험한 평균값을 측정하였다.

2.2 성능측정 결과

그림 4-7과 4-8에 50만개와 100만개의 접수데이터에서 측정한 조회 소요시간 결과를 보였다. 조건을 이용한 순차탐색의 경우 대기리스트 환자 수에 관계없이 조회 소요시간이 거의 일정하게 나타난 반면에, 주키탐색의 경우 대기리스트 환자 수에 비례하여 조회 소요시간이 증가하였다. 그러나 주키탐색에 소요된 시간은 순차탐색에 매우 작을 뿐만 아니라, 누적 데이터 수가 증가할수록 그 차이에 더욱 커짐을 보였다.

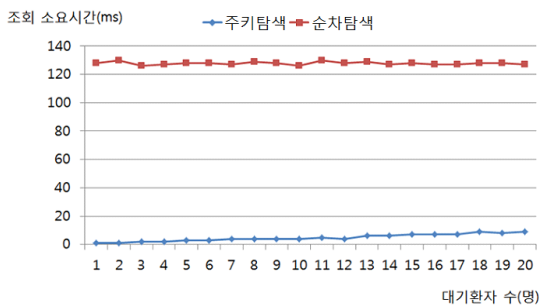

그림 4-7. 대기환자수별 조회시간 비교(접수데이터 50만개)
Fig. 4-7. Search Time Comparison on Waiting Patients(0.5 Million Reception Data)

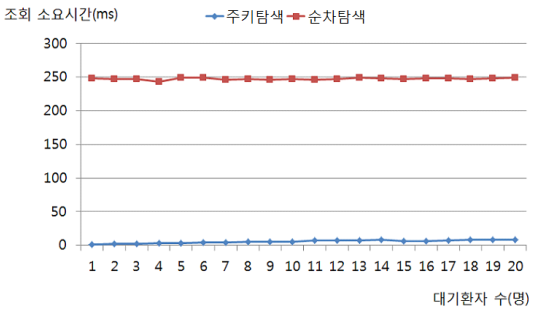

그림 4-8 대기환자수별 조회시간 비교(접수데이터 100만개)
Fig. 4-8. Search Time Comparison on Waiting Patients(1 Million Reception Data)

표 4-3에 대기환자 수에 따른 조회 소요시간을 누적데이터 수별로 보였다. 표 4-3의 행은 접수테이블의 누적 접수환자 수로서 10만개부터 100만개까지 10만개 단위로 나타내었고, 열은 대기환자 수로서 5명 단위로 보였다. 표 4-3의 실험결과는 접수번호를 전송하는 주키탐색이 조건에 의한 순차탐색에 비해 조회 소요시간이 크게 줄었으며, 누적 접수환자 수가 증가할수록 그 차이가 더욱 커짐을 보였다. 100만 누적 접수환자의 경우 주키탐색을 통해 20명의 대기환자를 조회하는 시간이 조건에 의한 순차탐색 보다 30배 이상 단축되었다.

표 4-3. 대기환자수별 조회시간 비교
Table 4-3. Time Comparison on Waiting Patients

	5		10		15		20	
	주키	순차	주키	순차	주키	순차	주키	순차
10	2	25	4	26	6	26	7	26
20	3	50	5	52	6	52	8	51
30	3	75	5	76	7	77	8	77
40	2	102	6	103	7	102	7	102
50	3	128	4	126	7	128	9	127
60	3	152	5	151	6	153	8	151
70	3	179	4	177	6	177	7	177
80	4	200	4	199	5	203	7	201
90	4	223	5	223	5	222	5	223
100	3	249	5	247	6	247	8	249

행 : 누적 접수환자 수(단위 : 만 명)
열 : 대기환자 수(단위 : 명) 및 탐색방법
데이터 : 조회소요 시간(단위 : ms)

표 4-3의 실험결과에서 대기환자 수가 20명일 때의 주키탐색과 순차탐색 시의 조회 소요시간을 누적 접수환자 수와 대비하여 그림 4-9에 정리하였다. 순차탐색은 누적환자 수에 비례해서 조회시간이 증가하는데 반해 주키탐색은 이와 무관하게 일정함을 보였다.

마지막으로 요청/응답과 웹소켓 방식에 의한 대기환자리스트 페이지 갱신에 소요된 시간을 측정한 결과를 그림 4-10에 보였다. 웹소켓 방식은 실시간 갱신이 이루어지므로 사용자요청 시에 대기환자리스트 페이지 전체를 갱신하는 요청/응답 방식보다 페이지 갱신 소요시간이 크게 줄었음을 보였다.

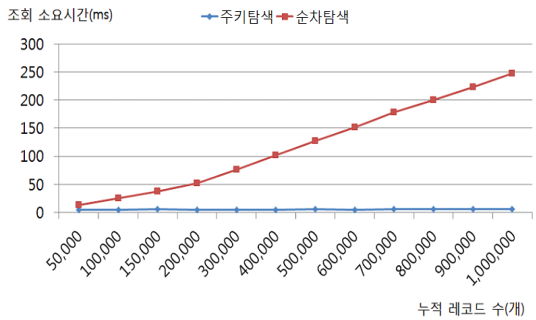

그림 4-9. 주키탐색과 순차탐색 시간 비교
Fig. 4-9. Time Comparison between Primary Key and Sequential Searches

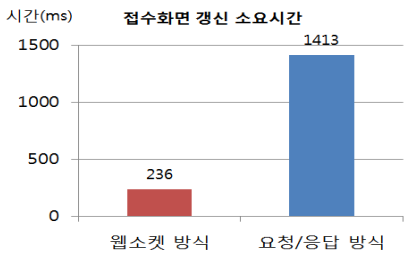

그림 4-10. 대기환자리스트 갱신 시간 비교
Fig. 4-10. Patient Waiting List Update Time Comparison

V. 결론

병원에서 외래접수 업무는 특정 시각대에 집중적으로 발생함에도 불구하고, 기존의 병원 접수시스템에서는 원무과에 새로 환자가 접수했다는 사실을 해당 진료과에 효과적으로 알려줄 수 있는 기술적 방식이 존재하지 않는다. 따라서 해당 진료과는 환자접수 상태를 알기 위해서는 일일이 수작업이나 타이머 방식에 의해 병원접수 데이터베이스를 조회해야 한다. 이러한 반복적인 과정은 번거로울 뿐만 아니라 데이터베이스 전체를 진료과 및 일시 등을 조건으로 하여 순차탐색해야 하기 때문에 방대한 양의 접수정보를 가진 병원정보시스템에서 성능저하의 큰 요인이 되고 있다.

본 논문은 병원 접수시스템의 효율을 향상시키기 위하여 웹소켓을 이용한 주키전송 일괄처리시스템과 접수정보전송에 의한 실시간시스템을 개발하였고, 접수 데이터베이스의 조회 성능을 순차탐색 일괄처리 접수시스템과 비교 측정하였다. 접수 조회에 있어서 주키전송 일괄처리시스템은 조건으로 순차탐색하는 방법 보다 조회 시간을 크게 단축시킬 수 있었다. 특히, 순차탐색은 누적 접수환자 수에 비례하여 조회 시간이 증가하는데 반하여, 주키전송 탐색은 접수환자 수에 무관한 빠른 조회속도를 유지하였다. 또한, 접수정보전송 실시간시스템은 데이터베이스를 조회하지 않고도 기존의 요청/응답 접수시스템의 대기환자 갱신시간을 효과적으로 향상시켰다.

참고문헌

〔1〕 Young-Mee Ryu, Il Ryu, Sora Kim, "Determinants of Hospital Nurses' Satisfaction on Hospital Information System(HIS): Focused on Perceived HIS Quality, Individual and Organizational Characteristics", The Journal of the Korea Contents Association, Vol. 13, No. 6, pp.438-449, 2013.

〔2〕 Hwa Young Jung, Jae Wook Park, Yong Kyu Lee, "A Context-Aware Treatment Guidance System", Journal of The Korea Society of Computer and Information, Vol. 19, No. 1, pp.141-148, 2014.

〔3〕 Yeon Wook Kang, "Donga University Hospital HIS 'DREAM' Open", Medical Today, *http://www.mdtoday.co.kr/mdtoday/index.html?no=234474*, 2013.

〔4〕 Dong-Il Cho, Sung-Yul Rhew, "Design and Implementation of Event Based Message Exchange Architecture between Servers for Server Push", Journal of Internet Computing and Services, Vol. 12, No. 4, pp.181-194, 2011.

〔5〕 Kiman Yang, "Performance analysis of asynchronous data communication using websocket of HTML5", Master Thesis of Graduate School of Industry, Kangwon National University, 2013.

〔6〕 Kung Lan Nam, "Response speed analysis of asynchronism WebScript", Master Thesis of Graduate School of Information, Chung-Ang University, 2012.

〔7〕 Wikipedia (2014), "Push technology", *http://en.wikipedia.org/wiki/Push_technology*.

〔8〕 Peter Lubbers & Frank Greco(2013), "HTML5 Web Sockets: A Quantum Leap in Scalability for the Web", *www.websocket.org/quantum.html*.

저 자 소 개

손 만 근

현 재: 순천향대학교 의료IT공학과

관심분야: 의료정보시스템, 웹 프로그래밍

Email : sonmg0416@gmail.com

박 기 성

현 재: 순천향대학교 의료IT공학과

관심분야: 웹 표준, 비 동기 통신

Email : kpw_believer@naver.com

공 용 해

1982: 연세대학교 전자공학과 공학사

1986: 뉴욕대학교(폴리텍) 컴퓨터과학과 공학석사

1990: 뉴욕대학교(폴리텍) 컴퓨터과학과 공학박사

현 재: 순천향대학교 의료IT공학과 교수

관심분야: 의료정보시스템, 시맨틱웹

Email : yhkong@sch.ac.kr