TCP/IP 소켓통신에서 대용량 스트링 데이터의 전송 속도를 높이기 위한 송수신 모델 설계 및 구현

강동조* · 박현주**

A design and implementation of transmit/receive model to speed up the transmission
of large string-data sets in TCP/IP socket communication

Kang Dong-Jo* · Park Hyun-Ju**

이 논문은 한국연구재단(과제번호 : 2012-0090)의 지원을 받아 수행된 연구임

요 약

TCP/IP소켓 통신을 활용하여 데이터를 송수신하는 송수신 모델에서 데이터의 크기가 작고 데이터 전송 요청이 빈번하지 않을 경우 서버와 클라이언트 간 통신 속도의 중요성은 부각되지 않지만 오늘날 대용량 데이터에 대한 전송 요청과 빈번한 데이터 전송 요청에서 송수신 모델의 통신 속도에 대한 중요성이 부각되고 있다. 본 논문은 대용량의 데이터를 전송하는 서버의 전송 구조와 데이터를 수신하는 클라이언트의 수신 구조를 변경하여 멀티 코어(이하 CMP : ChipMulti Processor) 환경에서 데이터 전송 속도에 대한 성능향상을 기대할 수 있는 보다 효율적인 TCP/IP 송수신 모델을 제안한다.

ABSTRACT

In the model Utilizing the TCP / IP socket communication to transmit and receive data, if the size of data is small and if data-transmission aren't frequently requested, the importance of communication speed between a server and a client isn't emphasized. But nowadays, it has emerged for large amounts of data transfer requests and frequent data transfer request. This paper propose the TCP/IP communication model that can be improved the data transfer rate in multi-core environment by changing the receiving structure of the client to receive large amounts of data and the transmission structure of the server to send large amounts of data.

키워드

TCP/IP소켓통신, 멀티쓰레드, 다중 소켓, 멀티코어

Key word

TCP/IP socket communication, Muti-Thread, Muti-socket, ChipMulti Processor

---

* 준회원 : 한밭대학교 전파공학과 석사(교신저자, hkhayona@naver.com)
** 정회원 : 한밭대학교 전파공학과 교수

접수일자 : 2012. 12. 20
심사완료일자 : 2013. 03. 04

I. 서 론

정보화 사회의 도래로 정보는 단순히 지식이 아닌 부가가치를 창출하는 하나의 자원이 되어 가고 있다. 부가가치를 창출하는 자원으로 사용하기 위해서는 다량의 정보에서 가치 있는 데이터를 검출하고 분류하는 시스템이 필요하며, 이러한 시스템은 데이터베이스와 연동하여 동작하는 규모가 큰 시스템이다. 하지만 이런 규모가 큰 시스템을 운용하는 것은 시스템을 운용하는 전문가의 필요와 시스템을 유지하기 위한 유지보수 비용 때문에 값비싼 비용이 든다. 때문에 데이터를 관리하는 시스템을 서버로 두고, 정보를 필요로 하는 각각의 클라이언트는 데이터가 필요할 때 서버에 요청하여 데이터를 수신 받는 구조가 일반적이다.

그림 1. 일반적인 송수신 모델
Fig. 1 General transmit/receive model

하지만 정보의 홍수와 요구되는 데이터 크기가 대용량화되면서 기존 TCP/IP 송수신 방식으로는 서버와 클라이언트 간에 효율적으로 데이터를 송수신할 수 없으며 데이터의 송수신에 새로운 패러다임을 필요함이 나타나고 있다. 특히, 데이터를 요청한 클라이언트는 서버로부터 데이터를 수신하는 동안에는 요청한 데이터를 활용하여 수행해야 할 작업은 필연적으로 작업 지연과 대기시간(latency)이 발생한다. 만약, 수신하는 데이터가 대용량일 경우, 작업의 지연은 더욱 길어질 수밖에 없다.

TCP 방식보다 송수신 속도가 빠른 UDP 방식이 존재하지만 UDP방식은 TCP방식에서 수행하는 흐름제어를 수행하지 않기 때문에 데이터 전송에 신뢰성이 떨어지는 단점이 있다[1].

본 논문은 대용량의 데이터를 전송하는 서버의 전송 구조와 데이터를 수신하는 클라이언트의 수신 구조를 변경하여 멀티 코어(이하 CMP : Chip Multi

Processor) 환경에서 데이터 전송 속도의 성능향상을 기대할 수 있는 TCP/IP 통신을 활용한 송수신 모델을 제안한다.

본 논문의 2장에서는 사전 지식이 될 수 있는 개념들을 살펴본다. 또한 본 논문이 지향하는 목표를 이루기 위해 송수신 모델이라는 주제로 3장에서는 설계관점으로 살펴보고, 4장에서는 구현관점에서 살펴본다. 5장에서는 제안한 TCP/IP통신에서 대용량 데이터의 전송 속도를 높이기 위한 송수신 모델에 대한 성능분석 및 평가를 한다. 6장에서는 제시하는 TCP/IP통신의 송수신 모델을 고찰하고 향후연구에 대하여 설명하고 결론을 맺는다.

II. 관련연구

2.1. TCP 소켓의 입출력 버퍼

TCP 소켓을 통해서 두 호스트가 연결되면, 상호간에 데이터의 송수신이 가능한 상태가 된다. 즉, 각각의 호스트는 [그림 2]과 같이 각각 입력버퍼와 출력버퍼를 갖는다.

그림 2. TCP 소켓의 입출력 버퍼 모델
Fig. 2 TCP socket's input and output buffer model

TCP 통신의 데이터 송수신은 데이터의 경계가 없으므로, 위에서 언급한 것처럼 TCP는 데이터의 흐름제어를 수행한다. 이때 TCP는 슬라이딩 윈도우(Sliding Window)라는 프로토콜을 활용한다. 슬라이딩 윈도우는 각 호스트의 입출력버퍼의 상태를 확인하여 수신 측의 입력버퍼가 수신 가능한 크기의 데이터만을 전송하고 나머지 데이터는 출력버퍼에 저장해 두었다가 수신 측 입력버퍼가 비워지면 이어서 데이터를 전송한다. 이는

신뢰성있는 데이터 송수신을 수행하지만 수신측에서 수신 데이터에 대한 처리가 신속하지 않거나 입력 버퍼의 저장 공간이 작으면 TCP통신이 지연될 수 있음을 보여준다[2].

이에 본 논문은 TCP통신의 잠재적인 지연을 발생시킬 수 있는 수신측의 수신 데이터 처리 속도를 향상시켜 송수신 속도를 높이기 위한 방법을 제안한다.

III. 송수신 모델 설계 및 구현

[그림 3]은 제안하는 송수신 모델의 전체 개념도이다.

그림 3. 제안하는 모델의 전체 개요도
Fig. 3 The whole overview of the proposed model

제안하는 송수신 모델은 크게 두 부분으로 나눌 수 있다. 첫 번째는 서버 부분으로 대용량의 데이터를 일정한 크기로 조각내어 패킷으로 만들어 전송하는 부분으로 하나의 클라이언트에 속해 있는 여러 개의 소켓의 연결 요청을 수락하고 연결한다.

두 번째는 클라이언트 부분으로 패킷형태로 전송된 데이터를 패킷의 헤더부분을 검증 및 분석하여 데이터를 수신하는 부분으로 기존처럼 하나의 소켓과 하나의 쓰레드로만 데이터를 수신하는 것이 아니라 여러 개의 소켓을 자신의 스택영역에 두고 해당 소켓으로 송신측과 연결하여 데이터를 수신하는 여러 개의 쓰레드를 생성하여 데이터를 수신한다.

서버와 클라이언트 간 1대 1 통신이지만, 서버 소켓과 클라이언트 소켓은 연결된 하나의 통로로만 데이터를 전송하는 것이 아니라 서버와 클라이언트에서 다수의 소켓을 열어 여러 개의 통로를 만들어 데이터를 전송한다.

3.1. 전송 모델 설계 및 구현

일반적인 전송 모델은 하나의 쓰레드로 패킷생성 및 전송을 수행한다. 패킷생성은 부하가 크지 않은 동작이지만 IPC자원인 소켓을 활용하여 패킷을 전송하는 작업은 부하가 큰 동작이다. 또한 네트워크 환경의 영향으로 패킷전송을 위해 호출한 시스템 콜의 반환이 지연될 수 있다. 이는 전송작업의 작업지연을 증가시키는 요인이 될 수 있으며, 전송작업의 속도를 감소시키는 요인이다[3][4][5]. 이에 제안하는 전송 모델은 패킷을 생성하는 부분과 패킷을 전송하는 부분으로 분리하여 작업지연을 감소시켜 전송작업의 속도 증가를 목표로 한다.

3.1.1. 데이터 전송 모델 구성

[그림 4]는 제안하는 송수신모델의 전체 구조이다.

그림 4. 제안하는 송수신 모델의 전체 구조
Fig. 4 The whole structure of the proposed transmission model

전송 모델은 크게 소켓관리유닛(socket management unit), 패킷생성유닛(packet creation unit), 패킷전송유닛(packet-send unit)으로 나눌 수 있다. 첫째로, 소켓관리유닛은 TCP 소켓을 생성하고 관리하며 수신측과 연결을 수행한다. 둘째로, 패킷생성유닛은 대용량의 데이터를

분해하여 분해된 데이터에 메시지 식별 번호(Message Number)와 데이터의 순서를 나타내는 순서번호(Sequence Number)로 구성된 헤더를 데이터 앞에 첨부하여 패킷을 생성한다. 셋째로, 패킷전송유닛은 TCP/IP를 활용하여 생성된 패킷을 수신측에 전송한다. 패킷을 전송할 때는 해당 수신측에서 연결 요청하여 연결된 모든 소켓에 차례대로 한 패킷씩 전송한다. Queue는 패킷 생성유닛에서 생성한 패킷을 저장하고 패킷전송유닛에서 패킷을 추출하는 저장공간으로 두 개의 쓰레드가 공동으로 작업을 수행한다.

3.2. 수신 모델 설계 및 구현

일반적인 수신모델은 하나의 소켓을 생성하고 생성한 소켓을 활용하여 송신측과 연결하여 연결된 하나의 통로로 데이터를 수신한다. 하나의 통로로 데이터를 수신할 경우, 수신측에서 수신한 데이터의 처리를 신속하게 처리하지 못하여 소켓의 수신버퍼가 가득 찬 경우, TCP 프로토콜의 흐름제어에 의해 데이터 전송의 지연이 발생된다. 이에 본 논문에서는 송신측과 연결된 수신측의 소켓 수를 증가시켜 수신버퍼가 가득 차 TCP 프로토콜의 흐름제어에 의해 데이터 전송의 지연 발생을 줄이고 각 소켓에 수신된 데이터를 처리하는 쓰레드를 생성하여 수신 데이터에 대한 처리속도를 증가시켜 전송 속도를 향상하는 것을 목표로 한다.

3.2.1. 데이터 전송 모델 구성

[그림 5]는 제안하는 수신모델의 전체구조이다. 수신 모델도 전송 모델과 같이 크게 세 부분으로 나눌 수 있다. 첫째로, 패킷수신유닛(packet-receive unit)은 TCP/IP를 활용하여 패킷을 수신하는 역할을 담당하는 부분으로 패킷 수신에 사용되는 쓰레드를 생성한다. 또한 생성된 쓰레드들은 데이터 수신에 사용될 소켓을 생성하며, 전송 측에 연결 요청을 수행한다. 둘째로, 패킷분석유닛(packet analysis unit)은 수신한 패킷을 분석하는 역할을 수행한다. 먼저 수신한 패킷이 자신이 요구한 데이터가 담긴 패킷이 맞는지 검사하고 다음으로 패킷에서 순서번호(Sequence Number)와 데이터를 추출한다.

그림 5. 제안하는 수신 모델의 전체 구조
Fig. 5 The whole structure of the proposed receiver model

마지막으로 오프셋추출유닛(Offset-position return unit)와 데이터저장유닛(data save unit)이다. 오프셋추출유닛은 패킷에서 추출한 순서번호를 사용하여 데이터가 저장될 파일안의 특정한 위치(이하 offset)를 반환한다. 만약 순서번호가 n 이고 패킷의 데이터의 사이즈가 s [byte]라면 offset은 식(1)을 활용하여 구할 수 있다.

$$offset = n \times s \quad \text{식(1)}$$

데이터저장유닛은 반환된 offset을 매개변수로 갖는 pwrite() 함수를 사용하여 데이터를 저장한다.

제안하는 수신모델은 다수의 쓰레드를 생성하여 데이터를 수신하며, 수신한 데이터를 원래의 데이터로 조합한다. 이를 위해 본 논문에서는 [그림 6]과 같은 데이터 수신 쓰레드 모델을 구성한다.

그림 6. 데이터 수신 쓰레드 모델
Fig. 6 The thread model receiving data

데이터를 수신하는 쓰레드의 생성 시기는 수신측에서 수신할 데이터가 존재할 경우, 쓰레드를 생성하는 방식을 선택하였다. 쓰레드를 생성하는 작업은 운영체제(이하 OS : Operating System)에 부담을 주는 작업이지만 데이터를 수신하는 경우에만 사용하는 쓰레드를 미리 생성해 두는 것은 한정적인 CPU 자원을 낭비하는 것이라 판단되었다. [그림 6]에서 확인할 수 있듯이 쓰레드 생성 시, TCP/IP 통신에 사용되는 소켓 또한 생성되며, 생성된 각 쓰레드는 자신의 스택 영역에 해당 소켓에 대한 정보를 가지고 있다. 각각의 쓰레드에서 생성된 소켓은 송신측과 연결된 연결 통로를 구축하며 해당 통로로 송신측과 통신한다. 시스템의 프로세스에서 생성된 쓰레드는 스택영역만 독립적으로 소유하며 나머지 데이터영역과 힙영역을 [그림 6]처럼 공유한다. 때문에 데이터영역에 수신한 데이터를 저장할 공유 영역의 파일 디스크립터를 위치시켜 모든 쓰레드가 공유할 수 있게 하였다.

IV. 성능 분석

4.1. 실험 환경

성능 테스트 실험환경은 송수신 모델의 장단점 및 차이점을 명확하게 확인할 수 있는 시나리오를 선정하여야 하므로 80MB이상의 파일을 사용한다. 송수신 시간을 측정하기 위해서 일반적인 모델의 측정시점은 소켓을 생성하기 전 부분에서 측정하며 제안하는 모델의 측정시점은 쓰레드를 생성하기 전 부분에서 측정한다. 송

수신 시간의 측정종료시점은 일반적인 모델에서는 소켓을 닫은 후 측정하며, 제안하는 모델은 생성한 모든 쓰레드가 종료한 후 측정한다.

[표 1]과 [표 2], [표 3]에서 테스트에서 사용한 공통된 사항과 수신버퍼 크기, 테스트 환경에 대해 기술하였다.

표 1. 테스트 공통 사항
Table. 1 The common

	공통 사항
테스트 파일	89,109,000byte string 파일
실험 횟수	20
측정 단위	sec

표 2. 실험을 수행한 수신 버퍼 크기
Table. 2 Performed experiment for the size of the receive buffer

수신버퍼크기	설명
87,380[Byte]	TCP소켓의 일반적인 수신버퍼 크기
32,768[Byte]	TCP소켓의 최소수신버퍼 크기

표 3. 성능 테스트 환경
Table. 3 The performance-test environment

	송신측	수신측
테스트 환경	VMware workstation	VMware workstation
운영체제(OS)	fedora8	fedora8
코어 수	1 개	2 개
주기억장치 (RAM)	1GB	4GB

[그림 7]은 송수신 모델의 성능 테스트를 위한 네트워크 환경이다. 성능 평가 네트워크 환경은 외부 네트워크 망과 연결되지 않은 서버(송신측)와 공유기(라우터) 클라이언트(수신측)로 이루어진 사설망을 구축하였다. [그림 7]과 같이 네트워크 환경을 구성한 이유는 네트워크 환경의 영향을 최소로 하여 송수신 모델의 성능을 평가하기 위해 선정하였다.

그림 7. 성능 테스트 네트워크 환경
Fig. 7 The performance-test-network environment

4.2. 동작검증

데이터 송수신 모델에 대한 성능 평가 지표로 데이터 송수신의 수행 속도뿐만 아니라 데이터 전송의 신뢰성 또한 중요하다. 본 논문에서는 데이터 전송의 신뢰성을 검증하고자 제안하는 모델의 동작 검증을 수행하였다.

[그림 8]는 수신측에서 수신한 데이터와 송신측에서 송신한 데이터에 대한 정보이다.

그림 8. 송/수신하는 파일의 정보 비교
Fig. 8 The information-comparisons sending / receiving a file

[그림 8]의 (b)는 송신측에서 수신측으로 전송한 데이터로 89,109,000byte 크기이다. 제시한 파일은 제안하는 송수신모델에 의해 전송되었으며, [그림 8]의 (a)에서 확인할 수 있듯이 동일한 크기의 데이터가 전송되었음을 확인할 수 있다.

[그림 9]은 두 파일을 비교하여 다른 부분을 출력해주는 유닉스에서 제공하는 diff 명령어를 확인하여 송신데이터와 수신 데이터를 비교한 그림이다.

그림 9. diff 명령어를 활용한 파일 비교
Fig. 9 The files comparisons using the diff command

송신한 파일과 수신한 파일이 동일하므로 출력창에 메시지가 출력되지 않았음을 확인할 수 있다.

동작 검증결과 제안하는 송수신 모델은 신뢰성 있는 송수신을 수행하였음이 확인되었다.

4.3. 성능테스트

[표 4]는 일반적인 송수신 모델의 송수신 수행시간을 나타낸 표이며, [표 5]와 [표 6]은 제안하는 송수신 모델의 송수신 수행시간을 나타낸 표이다.

[표 4], [표 5], [표 6]에서 확인할 수 있듯이 일반적인 송수신 모델과 제안하는 송수신 모델 모두에서 수신버퍼에 크기에 따라 성능의 차이가 발생한다.

표 4. 일반적인 송수신 모델의 수행시간
Table. 4 The execution time of General transmit / receive model

구분	수신버퍼 크기	
	87,380[Byte]	32,768[Byte]
	수행시간[sec]	
최소수행시간	27	42
최대수행시간	30	46
평균	28.01	43.25

[표 4]에서 확인할 수 있듯이 87,380byte 수신버퍼를 가진 일반적인 송수신 모델의 평균 수행 속도와 최소 수행 속도는 각각 28.01sec와 27sec로 측정되었으며, 32,768byte 수신버퍼를 가진 일반적인 송수신 모델의 평균 수행 속도와 최소 수행 속도는 각각 43.25sec와 42sec로 측정되었다. 이는 [표 5]과 [표 6]에서 제안하는 송수신 모델이 4개의 쓰레드로 동작할 경우, 수신버퍼의 크기가 각각 87,380byte, 32,768byte인 제안하는 송수신 모델의 평균 수행 속도와 최소 수행 속도인 23.55sec, 23sec와 26.3sec, 25sec와 비교하였을 때, 수신버퍼 크기가 87,380byte일 때, 평균 수행 속도는 4.46sec, 최소 수행

속도는 4sec의 차이를 발생하며, 수신버퍼 크기가 32,768byte일 때, 평균 수행 속도는 16.95sec, 최소 수행 속도는 17sec의 차이를 발생함을 확인할 수 있다.

표 5. 87,380byte크기의 수신버퍼에서 제안하는 모델의 수행시간

Table. 5 The execution time of the proposed model for the receive buffer size of 87,380 byte

구분	수신버퍼 크기 87,380[Byte]		
	쓰레드x2	쓰레드x3	쓰레드x4
	소켓 x2	소켓 x3	소켓 x4
	수행시간[sec]		
최소수행시간	23	23	23
최대수행시간	29	26	25
평 균	24.7	24.05	23.55

표 6. 32,768byte크기의 수신버퍼에서 제안하는 모델의 수행시간

Table. 6 The execution time of the proposed model for the receive buffer size of 32,768 byte

구분	수신버퍼 크기 32,768[Byte]		
	쓰레드x2	쓰레드x3	쓰레드x4
	소켓 x2	소켓 x3	소켓 x4
	수행시간[sec]		
최소수행시간	32	27	25
최대수행시간	34	29	29
평 균	33.2	28.05	26.3

[표 7]와 [표 8]은 일반적인 송수신 모델의 송수신 수행시간을 제안하는 송수신 모델과 비교하였을 때 송수신 수행시간의 성능향상은 퍼센트[%]로 나타낸 표이다.

표 7. 87,380byte크기의 수신버퍼에서의 성능평가
Table. 7 The performance evaluate for the receive buffer size of 87,380byte

구분	수신버퍼 크기 87,380[Byte]		
	쓰레드x2	쓰레드x3	쓰레드x4
	소켓 x2	소켓 x3	소켓 x4
	성능향상 [%]		
최소수행시간	14.81	14.81	14.81
최대수행시간	3.33	13.33	16.67
평 균	11.82	14.14	15.92

표 8. 32,768byte크기의 수신버퍼에서의 성능평가
Table. 8 The performance evaluate for the receive buffer size of 32,768 byte

구분	수신버퍼 크기 32,768[Byte]		
	쓰레드x2	쓰레드x3	쓰레드x4
	소켓 x2	소켓 x3	소켓 x4
	성능향상 [%]		
최소수행시간	23.81	35.71	40.48
최대수행시간	26.09	36.96	36.96
평 균	23.24	35.14	39.19

성능 테스트 결과 얻은 결과에서 확인할 수 있듯이, 일반적인 송수신 모델에서 대용량 데이터를 송수신하였을 때 보다 제안하는 송수신 모델에서 대용량 데이터를 송수신하였을 때 수신버퍼의 크기와 쓰레드의 개수에 상관없이 전송속도가 향상됨을 확인할 수 있다.

또한 제안하는 모델의 성능 평가 결과 [표 3]의 환경에서 수신측의 코어 개수가 2개 일 때, 쓰레드와 소켓을 4개 생성하여 수신하였을 경우, 성능이 가장 우수하게 나타났다. 이는 코어의 개수가 2개일 때, 쓰레드 4개를 생성하여 동작하였을 경우, 제안하는 송수신 모델에서 가장 성능향상에 이점이 있음을 확인하였다.

V. 결론 및 향후 연구

본 논문에서는 이러한 서버와 클라이언트 간 대용량 데이터의 통신 속도를 향상시키고 신뢰성 높은 송수신을 위하여 TCP/IP 통신을 활용한 송수신 모델을 제안하였다. 이를 위하여 전송 모델에서는 소켓 생성 및 연결을 위한 소켓관리유닛과 패킷을 생성하는 패킷생성유닛, 패킷을 전송하는 패킷전송유닛, 데이터 전송과 패킷을 생성하는 쓰레드 모델, 데이터 재조합을 위한 패킷구조를 구현하였다. 또한 수신 모델에서는 수신을 담당하는 쓰레드 생성, 소켓생성, 소켓 연결, 데이터 수신등의 작업을 하는 패킷수신유닛, 수신한 패킷을 검증하고 분석하는 패킷분석유닛, offset를 반환하는 오프셋추출유닛, 임계영역으로 보호 없이 데이터를 저장하는 데이터저장유닛을 구현하였다.

구현한 TCP/IP 통신을 활용한 송수신 모델의 성능평가를 위해 통신 속도 향상을 테스트하기 위한 성능 테스트, 통신의 신뢰성을 검증하기 위해 동작 검증을 5장 성능 분석에서 수행하였으며, 그 결과로, 본 논문에서 제안하는 TCP/IP 통신을 활용한 송수신 모델은 일반적인 통신 모델과 비교하였을 때, 대용량 데이터의 송수신 속도가 수신측의 수신버퍼의 크기가 87,380byte에서는 쓰레드와 소켓의 개수가 2개일 경우 평균 11.82%, 3개일 경우 평균 14.14%, 4개일 경우 평균 15.92% 향상되었으며, 32,768byte에서는 쓰레드와 소켓의 개수가 2개일 경우 평균 23.24%, 3개일 경우 평균 35.14%, 4개일 경우 평균 39.19% 향상됨을 확인하였다. 또한 동작 검증을 통해 제안하는 송수신 모델의 신뢰성 또한 확인하였다.

참고문헌

- [1] 윤성우, “열혈 TCP/IP 소켓 프로그래밍”, 오렌지미디어, pp53-54, 2007.
- [2] 윤성우, “열혈 TCP/IP 소켓 프로그래밍”, 오렌지미디어, pp134-136, pp170-171, 2007.
- [3] 김선영, “Advanced 리눅스 네트워크 시스템 프로그래밍”, 가메출판사, 2nd Edition, pp378-382, 2012.
- [4] 로버트 러브, “리눅스 시스템 프로그래밍”, O'REILLY, 한빛미디어, pp170-173, 2007.
- [5] <http://jeix2.blog.me/80015314844>., 2012.
- [6] 임찬숙, “다중경로 환경의 네트워크 코딩에서의 TCP 성능개선 방안”, 한국인터넷방송통신학회논문지, 제 11권, 제6호, 2011.
- [7] 양정아, 김동화, 김영호, “Linux에서의 다중통신포트 장치 구현”, 한국정보과학회, 학술발표논문집 Vol. 24, 1997.

저자소개

강동조(Dong-Jo Kang)

2011 한밭대학교 전과공학과 졸업
(학사)

2011~현재 한밭대학교
정보통신전문대학원
전과공학과 재학 (석사과정)

※관심분야: 유/무선 통신 소프트웨어, etc.

박현주(Hyunju Park)

1990년 서울 시립대학교
전산통계학과 학사

1992년 서울대학교 대학원
전산과학과 석사

1997년 서울대학교 대학원 전산과학과 박사

1997년~현재 한밭대학교 전과공학과 교수

※관심분야: 데이터베이스, 무선통신소프트웨어