Journal of Digital Contents Society Vol. 8 No. 3 Sept. 2007(pp. 393-401)

Device Discovery using Feed Forward Neural Network in Mobile P2P Environment

Kihyeon Kwon*, Hyung-Gi Byun**, Namyong Kim***, Sangchoon Kim****, Hyung-Bong Lee*****

Abstract

P2P systems have gained a lot of research interests and popularity over the years and have the capability to unleash and distribute awesome amounts of computing power, storage and bandwidths currently languishing - often underutilized - within corporate enterprises and every Internet connected home in the world. Since there is no central control over resources or devices and no before hand information about the resources or devices, device discovery remains a substantial problem in P2P environment. In this paper, we cover some of the current solutions to this problem and then propose our feed forward neural network (FFNN) based solution for device discovery in mobile P2P environment. We implements feed forward neural network (FFNN) trained with back propagation (BP) algorithm for device discovery and show, how large computation task can be distributed among such devices using agent technology. It also shows the possibility to use our architecture in home networking where devices have less storage capacity.

Keywords : Peer-to-Peer, 디바이스 탐색, feed forward neural network, 역전파

1. Introduction

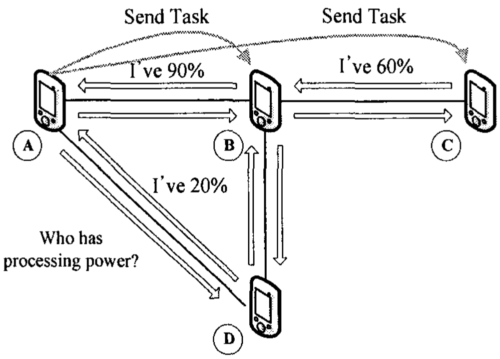
Peer-to-Peer which is commonly known as P2P, has been gripping the world wide web ra pidly making itself choice of many end users f or sharing resources, disseminating information and distributing tasks[1,2]. One main reason is due to economic slowdown where users are lo oking for getting maximum benefit from the h ardware they have. Some prominent advantage s of P2P systems are greater bandwidth, more computing power (storage, memory, CPU cycle s) available, and more people connected and m ore data generated. In its short period, P2P sy stems have overtaken client-server model due to its unique characteristic i.e. every networke d device acts as both client and server[1]. Las t decade saw development of various applicatio ns such as Napster[3], Gnutella[4], KaZaA[5] a nd JXTA[6] bringing P2P in limelight.

Furthermore, millions of users connecting to the Internet have started to work in group, in collaboration knowingly or unknowingly posses sing the possibility of becoming supercomputer s if integrated properly. It is believed that less than half of present computer processing powe r is used in real[1,2,7]. Hence, these powerful machines are not utilized to full capabilities an d have enough idle CPU cycles or storage cap acity for use. With the help of discovery, thos e dark matters of the Internet (unused CPU a nd storage) can be traced and used efficiently. But, discovery of available resources or device s in these environments, especially in mobile P 2P environment is a challenging task.

All the available discovery algorithms follow certain rules while forwarding any queries. W hen a device receives a query, it forwards it t o its neighbors or drops the query using certa in parameters like TTL(Time To Live), earlier replies or success rate which stop them to be optimal. The parameter dependency becomes t heir limitation. Hence it is necessary to minimi

※ 제일저자(First Author) : 권기현
접수일자:2007년07월02일, 심사완료:2007년07월13일
* 강원대학교 전자정보통신공학부
anyjava@empal.com
** 강원대학교 전자정보통신공학부
*** 강원대학교 전자정보통신공학부
**** 강원대학교 전자정보통신공학부
***** 강릉대학교 컴퓨터공학과

ze these parameters to allow zero configurability when applied to a real environment and/or there should be an efficient algorithm which can utilize various strategies at the same time. To overcome these limitations intelligent resource discovery technique is required. Hence we applied neural network based on feed forward neural network (FFNN) which learns by itself according to the situation in the given conditions and uses many combinations of strategies to locate resources. (Figure 1) shows, peer 'A' is how to find and select other peer by available resources and disperse the task to selected peers.



(Figure 1) Discovery in P2P

In this paper, we've implemented feed forward neural network[11] trained with back propagation(BP) algorithm[12] to discover devices in mobile P2P environment. FFNN uses various parameters as input values such as free CPU, the number of neighbors any device has and the number of hops etc. The next job is large computation with the help of agent technology dividing and distributing the task among capable devices.

The paper is structured as follows. Section 2 discusses the related works. Section 3 describes about neural network. Section 4 introduces our proposed architecture for device discovery where as section 5 describes the implementation and preliminary results. Finally, we have summarized the conclusions in section 6.

2. Related Work

There have been many approaches to discover P2P in the mobile environment. In Napster, a node sends request query to the centralized server which begins search with those nodes which are registered in it. After successful search, the file exchange occurs direct between nodes without any control from the server. The problem is in the central information storage which means single point failure, non-scalable and the requirement of central administration.

Gnutella algorithm propagates the request to all its neighbors until TTL (time to live) value. As search completes, the result is forwarded to their neighbors and finally to the requested node. But the flooding produces high overload to large number of peers and it doesn't scale too. There is also problem to define proper value for TTL.

In Random Walks, the query sender node sends k-query messages to those k-nodes which are selected in random basis. The queries are known as walkers and terminate either on success or failure. Though it decreases the number of messages or search results significantly [13,14] however, the result varies according to network topology and random choices.

It's modified version of BFS (Breadth First Search). Nodes keeps the update information about their neighbors in order to rank them which helps the node to forward the query to the selected neighbors that have returned the most results for similar queries. Though it scales well in accuracy and knowledge sharing and it also induces no overhead during nodes arrival/departure, but it produces large number of messages and shows no easy adaptation of resource deletions or peer departures.

In Directed BFS[15], each node keeps track of the success rates of earlier queries for parti cular resource which reflects the relative proba bility of this node's neighbor to be chosen as the next hop in a future request for that parti cular resource. The searching is based on ran dom walks forwarding the query to one of its neighbor with probability. It updates the indexi ng using feedback from the walkers. Directed BFS has feature of learning and induces zero overhead over the network at join or leave/up date.

In Local Indices[16], each node indexes the f iles stored at all nodes inside a certain periphe ral, i.e. within a certain radius. And, it can an swer on behalf of those nodes performing the search in BFS-like manner. But, in Routing In dices[17], documents are supposed to fall into a number of thematic categories and each nod e has information about approximate number o f documents from each category that can be r etrieved through each outgoing link. The forw ard process is similar to DFS (Depth First Se arch) and index maintenance requires flooding messages initiated from nodes that arrive or u pdate their collections.

The nodes that have no information about t he requested query forwards the query to all of its neighbors with certain probability in DR LP[18]. In case of found resource, the query ta kes the reverse path to the requester and regi sters the resource location and in the subsequ ent search, it contacts directly to the specific node.

3. Neural Network

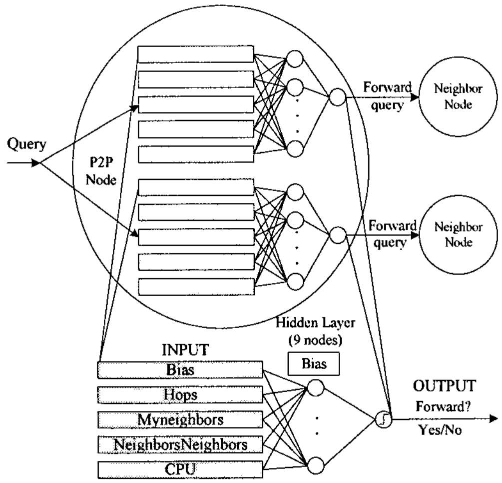
Breadth First Search (BFS) flooding algo-rithm is sends query to all neighbors. So, all resources in the network can be found, but network gets congested and there are lots of useless packets. But, neural network and evo-lution can be adapt its behavior to given environment. The neural network is used for deciding whether to pass the query further down the connection or not. And, evolution is used for breeding and finding out the best neural network in a large class of local search algorithms.

Artificial Neural Network (ANN) is an in-terconnected group of artificial or biological neurons. These neurons are organized as com-plex structure with the help of special con-nectors called synapses[8,9]. AAN is intelligent system that is similar and based on biological model of human brain. It operates on similar principle like a biological neuron where each incoming synapse of a neuron has a weight associated with it. The neural network is trained by adjusting weights between network elements and has a self learning capability, fault tolerant, and has been used in a broad range of applications, including: system identi-fication, pattern recognition, pattern completion, function approximation, optimization, prediction, automatic control[19]. ANN is potentially use-ful for studying the complex relationships be-tween inputs and outputs of a system[20]. There are many ANN models; one of the prominent is back propagation(BP)[20]. In this paper, three-layer feed forward neural net-work(FFNN) with sigmoidal function as acti-vation function in hidden layer followed by output layer is employed. The neural network is trained using BP algorithm. A momentum term is used in the BP algorithm to achieve a faster global convergence. A bias value is used to enable each neuron to fire hundred percent.

4. Architecture

4.1 Device Discovery Architecture using Feed Forward Neural Network

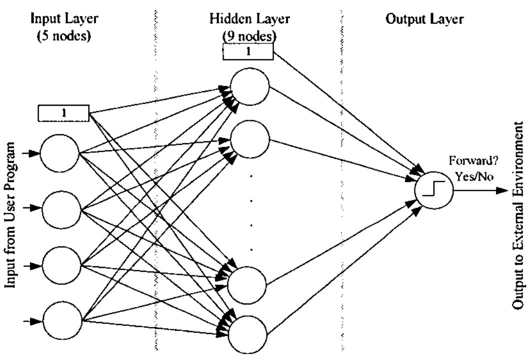
As in (Figure 2), we implemented a device discovery architecture is based on feed forwar d neural network (FFNN). Once a peer gets a ny query, it decides where to forward or not running FFNN for all individual peers. The peer is connected with two other peers (devices). It runs FFNN for both peers and decides whether to forward or not to any particular peer. It can forward both of them or one or none according to the result of FFNN.



(Figure 2) Query Processing of FFNN Architecture

4.2 Feed Forward Neural Network

Our proposed feed forward neural network architecture is as in (Figure 3). It has an input layer, a hidden layer and an output layer. The input layer is connected to hidden layer and hidden layer to output layer.



(Figure 3) Feed Forward Neural Network Structure

4.2.1 The Input Layer

The input layer is the conduit through which the external environment sends a pattern to the neural network. It should represent the condition for which we're training the network for. As in (Figure 2), we defined input values as Bias, Hops, Myneighbors, NeighborsNeighbors and CPU where Bias is always 1, Myneighbors is the number of neighbors any node has, NeighborsNeighbors is the number of nodes any node's particular neighbor has. CPU is the available CPU percentile of any particular node.

4.2.2 The Hidden Layer

The hardest job in neural network is to define the number of hidden layers. Since neural networks with two hidden layers can represent functions with any kind of shape and with one hidden layer is enough for many practical problems, we used one hidden layer. The number of hidden layer's neuron is very important part of deciding the overall performance of any neural network since this layer influences the output most. Using rule-of-thumb, we decided 9-neurons for the hidden layer.

4.2.3 The Output Layer

Since our system needs to decide whether to forward or not forward, we fixed output as a single value which gets either 1 (forward) or 0 (not forward).

4.2.4 Algorithm

The back propagation(BP) algorithm is one of the most important and widely used training methodologies for neural network. Learning takes place based upon mean squared error and gradient descent. And, BP makes it easy to find the networks error weight gradient for a given pattern. It is sometimes known as generalized Delta rule.

The steps of algorithm are as follows.

(1) Initialize weights
Each weight in the network initialized to some small random value

(2) For next pattern

① Perform a forward propagation step

$u_i = f(S_i)$

where $S_i = \sum^{j} w_{ij} u_j$ and $f(x) = \frac{1}{1+e^{-x}}$

First, net weighted sum S_i is calculated and activation u_i for each neurons using sigmoidal activation function.

② Perform backward propagation

$f'(x) = f(x)(1-f(x))$ or $f'(S_i) = u_i(1-u_i)$

if u_i is an output unit,

$\delta_i = u_i(1-u_i)(C_i - u_i)$

if u_i is a hidden unit,

$\delta_i = u_i(1-u_i)\sum^{h} \delta_h w_{hi}$

Error is calculated starting from outputs and propagated back to the hidden layer and input layer as above. C_i is the weighted sum of the errors.

③ Update weights

$w^*_{i,j} = w_{i,j} + \rho \delta_i u_j$

Weight update is done online immediately after the forward propagation as above. Momentum term was added to reduce the training time.

$w^*_{ij} = w_{ij} + \alpha \Delta w_{ij} + \rho \delta_i u_j$

where Δw_{ij} is the previous weight change. And, alpha is the momentum term.

Weight update is done online immediately after the forward propagation as above.

(3) Stop when total error is acceptable

① compute total error

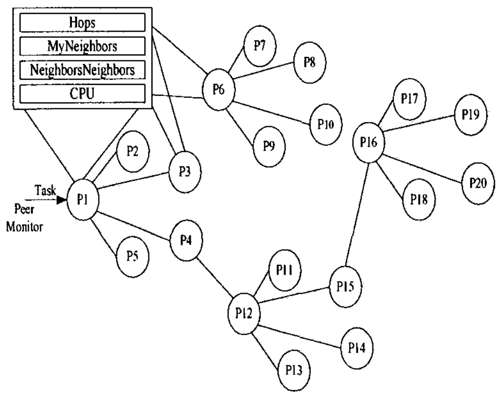
② if acceptable STOP, otherwise GO back STEP 2

Algorithm stops when the value of the error function has been sufficiently small.

5. Neural Network

5.1 Implementation Environment

We carried out experimental works with 20 embedded kits, called PXA250, to confirm the proposed device discovery architecture using FFNN.



(Figure 4) Experimental Topology
(Note: P means a peer.)

(Figure 4) shows experimental topology. <Table 1> shows the hardware specificatio n for PXA250. It was arranged in an ad-ho c network using wireless LAN.

<Table 1> Hardware Specification

Item	Description
Processor	Intel PXA250 400MHz
SDRAM	Samsung 64MB
Flash	Intel strata flash 32MB
Wireless LAN	WLI-USB-L11G
Display	LG TFT 6.4"(640*480)
RTC	RTC4513(Real Time Clock)
MMC, CF	1 Slot, 1 Slot

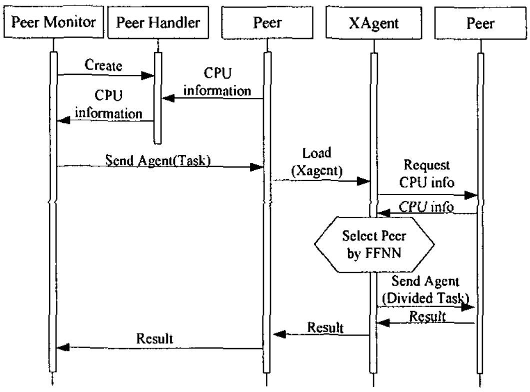
We used Java as developing language (JDK 1.3.1, JRE1.3.1) and Linux as operating system

(Kernel 2.4.18). The details for software specifi cation are expressed in <Table 2>.

<Table 2> Software Specification

Item	Description
O/S	Linux 2.4.18
Device Driver	CS9800 Ethernet, PCMCIA, CF, MMC
	Frame Buffer
	ADS7843 (Touch Screen)
File System	JFFS2, Ramdisk
GUI	Tiny X Server

5.2 Prototype



(Figure 5) State Transition Diagram

The prototype implementation consists of th e following components: (1) a PeerMonitor (P M), (2) an agent and load balancing mechanis m, and (3) a Feed Forward Neural Network.

(Figure 5) shows state transition diagram of the prototype.

5.2.1 Peer Monitor

Our architecture assumes that PM (PeerMon itor) works as a firing agent and shows its ne ighbors state like CPU percentile, IP and mem ory percentile etc. And each peer acts as clien t and server according to the situation at any time. It creates PeerHandler and gets CPU inf ormation from the peers and then, sends Agen t with certain task. That peer looks for neighb or peers and decides to whom the task should be sent according to their CPU information dis tributing the task among all available peers.

5.2.2 Agent and Load Balancing

An agent is used for distributing task with proper balancing algorithm, here we use divide and conquer. Agents are autonomous and responsible for sending task or returning results or gathering information.

5.2.3 Training Feed Forward Neural Network

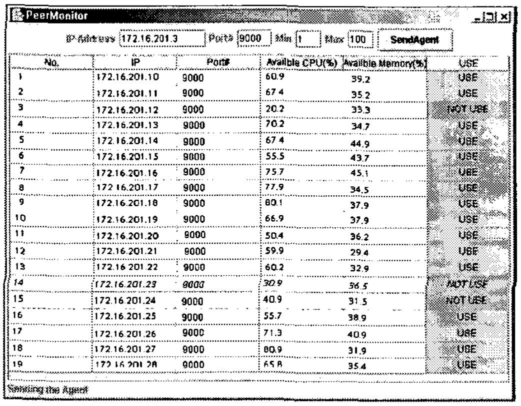
We formed four subnets from twenty embed ded devices in our laboratory, placing five in e ach subnet as in (Figure 4) and run FFNN tr aining with BP to decide whether its any parti cular neighbor has sufficient resource or not.

Let's suppose the peer 1 has a task. It need s to decide whether some of its neighbor can help him or not, for example Peer 1, Peer 2, P eer 3 and Peer 4 as in (Figure 4) First, Peer 1 secures the values for the parameter Hops, Myneighbors, NeighborsNeighbors and CPU in a set [Hops, Myneighbors, NeighborsNeighbor s, CPU] for each neighbor peers. Then, it feed these values to feed forward neural network a nd trains with backpropagation. In the case of Peer 1, since the query starts from Peer 1, so Hops value is zero, Myneighrbors value is 4 s ince it has four neighbors: Peer2, Peer 3, Peer 4, and Peer 5. And, the value of NeighborsNei ghbors and CPU are different for each of its Myneighbors. Let's take the case of Peer 2, fo r whom Peer 1 gets NeighborsNeighbors value is 0 since Peer 2 has no neighbors and CPU value is the available CPU of Peer 2. Then, th ese values forms the set[0,.4,0,1] which is fed into the neural network for training.

Since Peer 1 has four neighbors, it first dec ides who are capable of computing the task. T hen, it sends the task to them using agent (e x. XAgent). Similarly, Peer 3 and Peer 4 can again divide the task since they have neighbor s. In this way, large computation is divided in to smaller ones and computed by various avail able peers in collaborative and distributed man ner where the unused CPU of each capable de vice (which FFNN decides) is used. As in (Fi gure 4) all the peers have parameters Hops, Myneighbors, NeighborsNeighbors, CPU and th eir values for each neighbor.

5.2.4 Preliminary Results

We have successfully implemented and got p reliminary results for (Figure 6) and (Figure 7)

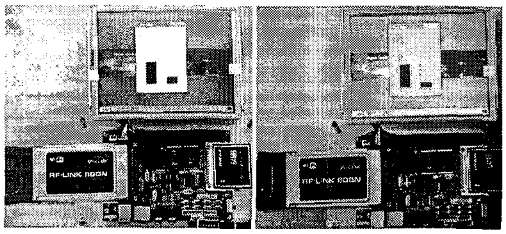


No.	IP	Port#	Availble CPU(%)	Availble Memory(%)	USE
1	172.16.201.10	9000	60.9	39.2	USE
2	172.16.201.11	9000	67.4	35.2	USE
3	172.16.201.12	9000	20.2	33.3	NOT USE
4	172.16.201.13	9000	70.2	34.7	USE
5	172.16.201.14	9000	67.4	44.9	USE
6	172.16.201.15	9000	55.5	43.7	USE
7	172.16.201.16	9000	75.7	45.1	USE
8	172.16.201.17	9000	77.9	34.5	USE
9	172.16.201.18	9000	80.1	37.9	USE
10	172.16.201.19	9000	66.9	37.9	USE
11	172.16.201.20	9000	50.4	36.2	USE
12	172.16.201.21	9000	59.9	29.4	USE
13	172.16.201.22	9000	60.2	32.9	USE
14	172.16.201.23	9000	30.9	36.5	NOT USE
15	172.16.201.24	9000	40.9	31.5	NOT USE
16	172.16.201.25	9000	55.7	38.9	USE
17	172.16.201.26	9000	71.3	40.9	USE
18	172.16.201.27	9000	80.9	31.9	USE
19	172.16.201.28	9000	65.8	35.4	USE

(Figure 6) PeerMonitor and Peer Status

(Figure 6) shows the state of each peer ava liable resource(CPU, memory) and status whet her the peer is working or not, and (Figure 7) is the experimental display of PXA250 embedd ed kit. We haven't yet made any performance comparison with other present technologies, bu t it showed high potentiality for the dynamic environment like P2P and could be an alternati ve. We are still working on this project and e xpecting to come up with further concrete adv antages.

(Figure 7) was the execution snapshot in ou r laboratory.



(Figure 7) Execution Snapshot

6. Conclusion

The Internet has thousands of computers connected and more than half of them are using less than half of their actual power (CPU, storage, Memory). P2P has brought great change in distributing computing leaving traditional client-server model in jeopardy. It utilizes the unused processing power. However the problem of device discovery remains a substantial threat on is development, existence. This work showed new direction to solve this credential problem of discovery using intelligent mechanism, neural network. Feed Forward Neural Network (FFNN) trained with back progation(BP) was used to discover the efficient devices from ocean of connected devices. Then, we distributed a large computation task using agent technology among those capable devices. This work also showed how to utilize the unused resources for handling any task. We implemented rudimentary neural network based architecture and achieved the feat to share or to lease the computing power/resources. Hence, P2P computing seems to have the potential to offer a better and intelligent solution in combination with neural network in device discovery. Although other techniques may prove accurate at the same task, the neural network seems to be suitable and sufficiently accurate choice.

Since the work is in preliminary stage, it needs to invest more research time to cement its legitimacy. Future works will cover performance evaluation and advantages over other technologies.

참 고 문 헌

[1] Clay Shirky, "What is P2P and What isn't?", O'Reilly Network., 2000
[2] A. Acharya, G. Edijlali, and J. Saltz, "The Utility of Exploiting Idle Workstations for Parallel Computation", SIGMETRICS 1997.
[3] Napster, Inc., http://www.napster.com
[4] Gnutella Web Site, http://www.gnutella.com
[5] Kazaa Web Site, http://www.kazaa.com.
[6] Project JXTA web site, http://www.jxta.org
[7] M. Litzkow, M. Livny, M. W. Mutka, "Condor - A Hunter of Idle Workstations", In Proceedings of the 8th International Conference of Distributed Computing Systems, pages pp. 104-111, 1998.
[8] J. Hertz, A. Krogh and R. G. Palmer, "Introduction to the Theory of Neural Computer", Addison-Wesley, 1991.
[9] M. H. Hasson, "Fundamental of Artificial Neural Networks", MIT Press, 1995.
[10] J. Heaton, "Introduction to Neural Networks with Java", SAMS Publishing, 2003.
[11] D. McAuley, "BackPropagation Network: Learning by Example" http://www2.psy.uq.edu.au/~brainwav/Manual/BackProp.html 1997.
[12] S. Kurkovsky, Bhagyavati, "Modeling Computational Grid of Mobile Devices as a Multi-Agent System", International Conference on Artificial Intelligent, 2003.
[13] B. J. Kim, C. N. Yoon, S. K. Han, and H. Jeong, "Path Finding Strategies in Scale-Free Networks", Physical Review, 65, 2002.
[14] D. Tsoumakos and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks", In 3rd, IEEE Intl. Conference on P2P Computing 2003.
[15] B. Yang, H. Garcia-Molina, "Improving Search in Peer-to-Peer Networks", In ICDCS, 2002.
[16] A. Crespo, H. Garcia-Mollina, "Routing Indices for Peer-to-Peer Systems", ICDCS, July 2002.
[17] T. Lin, H. Wang, "Search Performance Analysis in Peer-to-Peer Networks", In Proceedings of the 3rd International Conference on P2P Computing (P2P'03), 2003.
[18] G. K. Venayagamoorthy, V. Moonasar, and K. Sandrasegaran, "Voice recognition using neural networks", IEEE , 29 - 32, 1998.
[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations of back-propagation Error.", Nature (London), 323, 533 - 536, 1986.
[20] D. Tsoumakos, N. Roussopoulos, "Analysis and Comparison of P2P Search Methods", Available at http://www.cs.umd.edu/Library /TRs/CS-TR-4539/CS-TR-4539.pdf
[21] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, "Load Balancing in Dynamic Structured P2P Systems", Infocom, IEEE, 2004.

권 기 현

1993년 : 강원대학교 전자계산학과 (이학사)
1995년 : 강원대학교 전자계산학과(이학석사)
2000년 : 강원대학교 컴퓨터과학과(이학박사)
2002년～현 재 : 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 조교수
관심분야 : 미들웨어, 임베디드시스템, 패턴인식

변 형 기

1984년 : 명지대학교 전기공학과 (공학사)
1990년 : Manchester University 전자전기공학과 MSc.
1995년 : Manchester University 계측공학과 Ph.D.
1996년～현 재 : 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 교수
관심분야 : 전자 후미각 시스템 및 패턴인식

김 남 용

1986년 : 연세대학교 전자공학과 (공학사)
1988년 : 연세대학교 전자공학과 (공학석사)
1991년 : 연세대학교 전자공학과 (공학박사)
1992년~1998년 : 관동대학교 전자통신공학과 부교수
1998년~현 재 : 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 교수
관심분야: Adaptive equalization, RBFN algorithm, odor sensing systems.

김 상 춘

1986년 : 한밭대학교 전자계산학과 (이학사)
1989년 : 청주대학교 전자계산학과 (공학석사)
1999년 : 충북대학교 전자계산학과 (이학박사)
2001년~현 재 : 강원대학교 공학대학 전자정보통신공학부 정보통신공학전공 부교수
관심분야 : 정보보호 시스템, 보안 네트워크, 암호 알고리즘

이 형 봉

1984년 : 서울대학교 계산통계학과 (이학사)
1986년 : 서울대학교 대학원 계산통계학(전산과학)과 (이학석사)
2002년 : 강원대학교 컴퓨터과학과(이학박사)
1986년~1994년 : LG전자 컴퓨터연구소 선임연구원
1997년~1999년 : 전자계산기조직응용, 전자계산기, 정보통신기술사
1999년~2004년 : 호남대학교 정보통신공학부 조교수
2004년~현 재 : 강릉대학교 컴퓨터공학과 조교수
관심분야 : 프로그램언어 및 보안, 운영체제, 알고리즘, 멀티미디어 통신