고성능 네트워크에서 병렬 전송 기술을 이용한 전송률 극대화 메커니즘

(A Maximum Mechanism of Data Transfer Rate using Parallel Transmission Technology on High Performance Network)

김 영 신 † 허 의 남 ††
(Young-Shin Kim) (Eui-Nam Huh)

요 약 NGI나 Internet2와 같은 프로젝트로 인해 인터넷 백본 속도가 상당히 높아졌음에도 불구하고, 분산된 응용 프로그램들은 고성능의 네트워크를 제대로 활용하지 못하고 있다. 이러한 현상이 발생하는 원인으로 표준 전송 프로토콜(TCP)을 들 수 있다. TCP는 안전성/신뢰성을 보장하기 위해 설계되어 있으나, 이로 인해 발생될 수 있는 성능 저하에 관한 문제는 고려되지 않았다. 이러한 문제를 해결하고자 여러 기술들이 연구되고 있으며, 그 중 병렬 전송 기술은 응용레벨에서 다중 스트림을 이용하여 데이타를 전송하는 기술로써, 호환성 문제까지 해결하고 있다. 최근 병렬 전송 기술을 연구하는 연구자들은 최적의 병렬연결 개수의 범위를 찾는데 연구의 초점을 맞추고 있다. 그러나 이러한 연구들에서는 최적의 병렬연결 개수를 실험을 통해 경험적으로 결정하고 있으며, 데이타를 전송하는 호스트의 성능이나 전송 거리는 고려하지 않고 있다. 따라서 본 논문에서는 호스트의 성능과 병렬 전송과의 관계, 전송 거리와 병렬 전송 관계를 분석하고, 그 결과를 토대로 효율적이면서 최대 전송 성능을 확보할 수 있는 최적의 병렬연결 개수 결정 메커니즘을 논의하고자 한다.

키워드 : 병렬전송기술, 네트워크 성능 향상, TCP 버퍼튜닝 기술

Abstract Even though Internet backbone speeds have increased in the last few years due to projects like Internet 2 and NGI, many high performance distributed applications are able to achieve only a small fraction of the available bandwidth. The cause of such problem is due to a character of TCP/IP. The primary goal of this protocol is reliable data transmission. Therefore high speed data transmission didn't be considered when TCP/IP is designed. Hence several researchers have been studied in order to solve the problem of TCP/IP. One of these research results, parallel transfer technique, solves this problem to use parallel TCP connections on application level. Additionally, this technique is compatibility. Recently, these researchers have been studied a mechanism to decide the number of parallel TCP connections. However, some researchers reported the number of parallel TCP connection base on only empirical results. Although hardware performance of host affects transmission rate, the hardware performance didn't be considered in their works. Hence, we collect all data related to transmission rate, such as hardware state information (cpu utilization, interrupt, context switch). Then, we analyzed collected data. And, we suggest a new mechanism determining number of parallel TCP connections for maximization of performance based on our analysis.

Key words : Parallel Transfer, High performance network, TCP buffer tuning

1. 서 론

지리적으로 분산된 자연 과학용 또는 공학용 응용 프로그램들, 또는 storage system들 간에 대용량의 데이타 전송은 빈번히 일어나고 있다. 이에 보다 빠른 데이타 전송이 요구되고 있다. 현재는 NGI나 Internet2와 같은 프로젝트로 인해 인터넷 백본 속도가 상당히 높아졌음에도 불구하고, 분산된 응용 프로그램들은 고성능의 네트워크를 제대로 활용하지 못하고 있다[1]. 이러한 현상이 발생하는 데에는 여러 가지 원인이 있겠으나, 그 중에 대두되고 있는 문제가 표준 전송 프로토콜(TCP)

· 이 연구는 2007년도 경희대학교 연구비지원에 의한 결과임(KHU-20070748)
† 학생회원 : 경희대학교 전자정보학부 연구원
yskim@icns.khu.ac.kr
†† 정 회 원 : 경희대학교 전자정보학부 교수
johnhuh@khu.ac.kr
논문접수 : 2007년 6월 13일
심사완료 : 2007년 8월 15일

에 있다. TCP는 높은 안전성과 신뢰성을 목표로 설계되어 있어, 데이타 전송 성능을 저하시키는 문제가 발생된다. 따라서 이러한 TCP의 문제점을 보완하고자 SABUL[2], XCP[3], DRS[4], ATBT[5], TBT-PLR[6], 병렬 전송과 같은 여러 기술들이 연구되고 있다. 그러나 SABUL, XCP, DRS, ATBT, TBT-PLR 등의 기술들은 커널 소스의 수정이나 커널 변수에 접근해야 하므로 호환성 문제를 가지고 있다. 하지만 병렬 전송 기술은 응용레벨에서 다중 스트림을 이용하여 데이타를 전송하므로 이러한 호환성 문제를 해결하고 있다. 이러한 이점으로 인해 GridFTP에서도 병렬 전송 기술을 이용하고 있으며[7], 최근 연구자들은 최적의 병렬연결 개수 범위를 찾는데 초점을 맞추어 연구를 진행하고 있다. Sivakumar[8]의 논문에서는 여러 차례의 실험을 거쳐 최적의 병렬연결 개수 범위는 8~12개라고 밝히고 있으며, Thomas[9]의 논문에서는 packet loss를 이용하여 최적의 병렬연결 개수를 결정하는 방법을 제시하고 있다[8,9]. 그러나 이들 연구에서는 최적의 병렬연결 개수는 경험적으로 결정된 값이며, 데이타를 전송하는 호스트의 성능과 전송 거리는 고려되지 않았다. 따라서 본 논문에서는 호스트의 성능과 병렬 전송과의 관계, 전송거리와 병렬 전송과의 관계를 분석하고, 그 결과를 토대로 효율적이면서 최대 전송 성능을 확보할 수 있는 최적의 병렬연결 개수 결정 메커니즘을 논의하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 병렬 전송 기술에 대해 알아보고, 3장에서는 여러 실험을 통해 수집한 데이타를 이용해 호스트 성능과 병렬 전송의 관계, 전송 거리와 병렬 전송의 관계를 분석한다. 그리고 4장에서는 제안하는 메커니즘에 대해 논의하며, 이어 5장에서는 성능평가를 실시한 후 마지막 6장에서는 결론과 향후 연구 과제에 대해 논의한다.

2. 관련연구

2.1 병렬 전송 기술의 원리

병렬 전송은 송신자와 수신자 간에 다중 TCP 연결을 설정하고, 설정된 TCP 연결 개수만큼 분할된 데이타를 각각의 연결을 통해 동시에 전송하는 기술이다. 이 기술은 고성능 원거리 네트워크에서 데이타 전송 능력을 향상시키며, 응용레벨에서 운영됨으로써 호환성 문제를 해결하고 있다. 이러한 이점 때문에 Globus의 GridFTP에서도 이 기술을 활용하고 있다[10]. 그림 1은 병렬 전송의 원리를 나타내고 있다. 다중 TCP 연결을 이용하여 데이타를 전송하는 상태와 단일 TCP 연결을 이용하여 데이타를 전송하는 상태를 비교하여 보여주고 있다[11,12].

그림 1의 (a)는 단일 TCP 연결로 데이타가 전송되고 있는 상태를 나타내고 있다. 수신 호스트로 전송된 데이타의 ack가 돌아오는 시간, 즉, RTT동안 다음 데이타는 전송되지 못하고 대기하는 모습을 볼 수 있다.

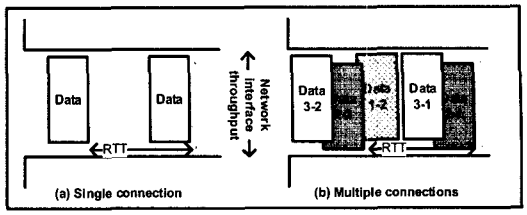

그림 1 단일 TCP 연결과 다중 TCP 연결의 비교

따라서 전송 거리가 멀어질수록 RTT는 길어지고 그만큼 대기하는 시간은 길어지게 된다. 그러나 그림 1의 (b)의 경우는 여러 개의 TCP 연결을 동시에 함으로써 한 TCP 연결에서 이미 전송된 데이타의 ack를 기다리는 동안 다른 TCP 연결들이 데이타를 전송하게 되므로 전송 호스트에서는 ack를 기다리는 대기 시간이 짧아지게 된다. 그러므로 전송거리가 멀어져 RTT가 길어지더라도 TCP 연결의 개수가 증가하게 되면 그만큼 전송대기 시간은 짧아지게 되고 전송 성능은 향상된다[13].

2.2 병렬 전송 기술의 연구 동향

Sivakumar[8]는 Psockets 라이브러리를 이용하여 실험한 결과 12개의 TCP 연결을 이용할 경우 전송률이 10Mbits/sec에서 약 75MBits/sec으로 증가하여 전송 능력이 향상되었음을 증명하고 있다. Thomas[9]는 네트워크 혼잡을 회피하면서 최대 처리량을 얻을 수 있는 TCP 연결의 개수를 결정하기 위해 다중 TCP 연결의 이론적인 모델을 제시하고 있다. 또한 GridFTP project에서는 병렬화 수준의 자동 조절 메커니즘을 연구하고 있다[7].

그러나 다중 TCP 연결의 설정과 운영은 호스트의 하드웨어 성능에 의해 많은 영향을 받음에도 불구하고 현재까지 연구에서는 고려되지 않고 있다. 즉, 실험을 통한 경험으로 TCP 연결 개수 범위를 선택하거나 packet loss와 RTT, MSS 요소들을 이용한 이론적인 모델을 제시하는데 그치고 있다.

3. 호스트 성능과 병렬 전송 기술과의 관계 분석

3.1 데이타 수집을 위한 실험 환경

그림 2는 데이타 수집을 위한 실험 환경을 나타내고 있다.

Host1과 Host2는 각각 1GMbps 스위치 허브(3Com Baseline Switch 2808)에 연결되어 있다. Host1은 데이타를 수신하는 역할을 수행하고, Host2~4는 데이타를 Host1로 송신한다. Host1에는 Nistnet[14]가 설치되어 있어 Host1과 Host2~4 간에 RTT를 조절한다. Host1

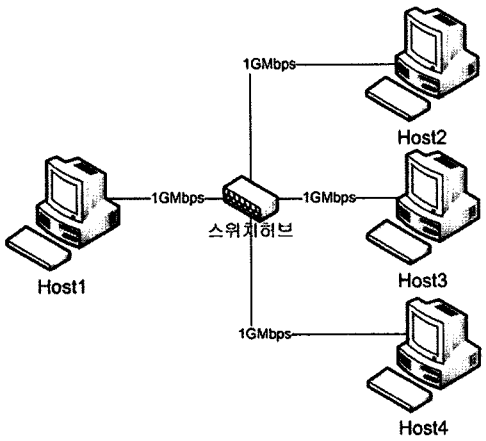

그림 2 실험 환경 1

과 Host2, Host1과 Host3, Host1과 Host4 간에 RTT는 20ms에서 100ms까지 5ms씩 증가시키며 설정하고, 각각의 경우마다 1Gbytes의 데이타를 다중 TCP 연결을 이용하여 전송하고 전송률과 시스템 데이타를 수집하였다. TCP 연결의 개수는 1, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40(이하 PC_i, i는 1~40)을 이용하였다. 수집하는 데이타는 전송률(Mbps), CPU 사용률(%), context switch 수, interrupt 수, buff에서 사용되고 있는 memory 양(KB), cache에서 사용되는 memory 양(KB)이며, 이들은 vmstat를 이용하여 1초 주기로 수집되었다. 실험 시 각 호스트에서는 실험 이외에 다른 목적으로 실행되는 프로세스는 없는 상태로 실험 전용으로 사용된다. 본 실험에서 사용된 호스트 Host1~Host4의 성능은 다음 표 1과 같다.

3.2 측정된 데이타 관계 분석

수집된 데이타들을 전송률과 병렬연결의 관계, CPU 시스템 사용률과 병렬연결의 관계, Context switch와 병렬연결과의 관계, Interrupt와 병렬연결의 관계, Memory 사용량과 병렬연결의 관계를 중심으로 정리하였다. 그러나 본 논문에서는 지면관계상 RTT 20ms, 60ms, 80ms, 100ms에서 측정된 데이타만을 표기하며, 가장 밀접한 관계를 보인 CPU 시스템 사용률과 병렬연결과의 관계만을 서술하고자 한다.

3.2.1 CPU 시스템 사용률과 병렬연결과의 관계 분석

3.2.1.1 Host1과 Host2의 경우

두 호스트 간에 다중 TCP 연결을 이용하여 데이타가 전송되는 동안 전송률과 CPU 시스템 사용률을 측정하고 그 값의 평균을 표 2와 3에 나타내고 있다.

표 2 Host1과 Host2 간의 전송률 (단위: Mbps)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	27.36	13.77	9.17	6.97	5.56
PC_4	110.1	55.42	37.23	27.92	22.27
PC_8	236.1	111.31	74.61	56.13	44.62
PC_{12}	210.6	170.16	111.16	83.24	66.88
PC_{16}	232.1	244.73	150.98	112.19	89.34
PC_{20}	179.79	226.64	196.93	141.65	113.16
PC_{24}	244.82	213.67	231.03	172.81	135.17
PC_{28}	206	220.87	217.14	211.34	159.14
PC_{32}	241.06	193.18	208.86	196.8	188.94
PC_{36}	229.95	226.14	198.9	204.15	227.73
PC_{40}	231.82	234.1	197.37	186.59	198.7

표 3 Host1과 Host2간의 CPU시스템사용률 (단위: %)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	9.41	4.93	3.33	2.54	2.05
PC_4	37.61	19.34	13.33	10.06	7.89
PC_8	78.64	38.72	25.90	19.79	15.78
PC_{12}	73.69	58.51	38.05	29.33	23.61
PC_{16}	82.2	86.03	53.22	39.54	31.24
PC_{20}	62.37	79.3	66.79	48.78	39.29
PC_{24}	84.56	74.64	79.19	59.42	46.31
PC_{28}	71.6	76.97	74.59	71.55	54.23
PC_{32}	81.31	68.86	73.03	70	64.93
PC_{36}	77.84	79.16	69.14	76.68	78.03
PC_{40}	78.22	81.47	69.69	64.47	70.56

표 1 실험 호스트 성능

구분 / 호스트명	CPU	Memory	LAN card	OS
Host1	Intel Celeron CPU 2.8GHz (cpuMHz 2795.256)	482180KB	32bit GIGA bit PCI LanCard GIGA-1000s	Fedora 4
Host2	Intel Celeron CPU 2.8GHz (cpuMHz 2795.256)	482180KB	32bit GIGA bit PCI LanCard GIGA-1000s	RedHat9 2.4.20-8
Host3	Intel Celeron CPU 1200MHz (cpuMHz 1196.035)	512292KB	32bit GIGA bit PCI LanCard GIGA-1000s	RedHat9 2.4.20-8
Host4	Intel Pentium III (cpuMHz 800.023)	61356KB	32bit GIGA bit PCI LanCard GIGA-1000s	RedHat9 2.4.20-8

표 2와 3을 비교해 보면, 전송률이 높아질수록 CPU 시스템 사용률도 높아지며, CPU 시스템 사용률이 최고치에 다다르면 전송률 또한 더 이상의 증가하지 않는 현상을 발견할 수 있다. 예를 들어, RTT 40ms에서 PC_1~PC_{18}까지 TCP 연결의 개수를 증가시키면 13.77 Mbps에서 244.73Mbps로 전송률이 약 18배 증가하게 되며, CPU 시스템 사용률도 같이 증가하는 현상을 볼 수 있다. 그러나 PC_{20}이상을 사용한 경우 전송률과 CPU 시스템 사용률 모두 뚜렷한 증가현상은 보이지 않고 있다.

그림 3은 표 3을 그래프로 나타낸 것이다.

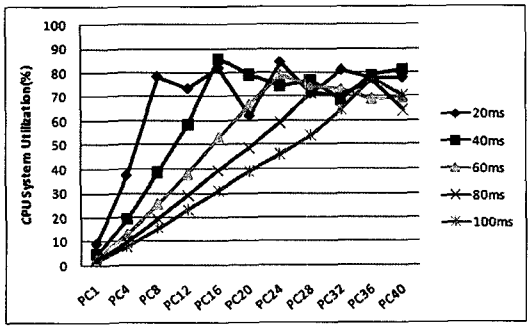

그림 3 Host1과 Host2 간에 CPU 시스템 사용률

각각의 RTT에서 TCP 연결 개수의 증가로 인해 CPU 시스템 사용률이 약 70%이상에 도달하게 되면 TCP 연결의 개수가 더 증가하더라도 CPU 시스템 사용률은 더 이상 증가하지 않음을 알 수 있다. 또한 RTT가 증가할수록 그래프의 기울기가 낮아지는 현상을 볼 수 있는데, 이러한 현상은 전송 거리가 멀어질수록 ACK가 그만큼 늦게 회신되고 그에 따라 다음 데이타가 전송되지 못하고 대기하게 되어 전송 지연이 발생되기 때문으로 판단할 수 있다. 따라서 RTT가 길어질수록 더 많은 연결을 설정하게 되면 한 연결에서 발생하는 지연시간동안 다른 연결에서 전송 작업을 수행할 수 있으므로 전송률은 증가하고 많은 TCP 연결을 사용하므로 CPU 시스템 사용률도 증가하게 된다.

3.2.1.2 Host1과 Host3의 경우

표 4는 병렬연결 개수에 따른 전송률을, 표 5는 CPU 시스템 사용률을 측정한 결과이다.

표 4와 5를 통해서 Host1과 Host3의 경우에도 대체로 CPU 사용률과 전송률이 비례하게 움직이고 있음을 알 수 있다. 그림 4는 표 5를 그래프로 나타낸 것이다.

Host3도 Host2와 같이 병렬연결의 개수가 증가할수록 CPU 시스템 사용률이 증가함을 알 수 있다. 또한 Host2와 같이 RTT가 증가할수록 그래프의 기울기는 감소되는 현상 역시 확인할 수 있다.

표 4 Host1과 Host3 간의 전송률 (단위: Mbps)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	23.8	11.95	7.98	6	4.8
PC_4	94.95	47.85	32.03	24.06	19.34
PC_8	186.49	95.62	63.98	48.12	38.62
PC_{12}	245.08	142.1	95.72	72.09	57.68
PC_{16}	259.93	183.94	127.17	95.84	76.47
PC_{20}	261.75	219.24	158.41	119.45	95.30
PC_{24}	266.61	251.88	188.86	142.92	114
PC_{28}	272.1	258.98	218.38	165.79	132.79
PC_{32}	269.05	265.69	242.18	187.75	151.52
PC_{36}	273.65	259.98	258.47	209.26	170.22
PC_{40}	275.05	269.27	258.81	229.65	188.46

표 5 Host1과 Host3 간의 CPU시스템사용률 (단위: %)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	6.43	3.29	2.26	1.8	1.34
PC_4	25.91	12.36	8.03	6.4	4.50
PC_8	47.91	24.01	15.40	12.82	9.45
PC_{12}	61.79	34.93	23.67	19.93	13.99
PC_{16}	65.42	52.70	31.45	26.37	19.78
PC_{20}	63.84	62.68	38.89	33.35	23.48
PC_{24}	67.53	63.82	45.96	40.02	29.31
PC_{28}	68.6	63.38	53.97	47.72	33.79
PC_{32}	67.77	66.26	59.35	52.84	38.93
PC_{36}	70.9	66.52	59.38	56.93	42.67
PC_{40}	69.17	68.33	64.69	60.58	47.7

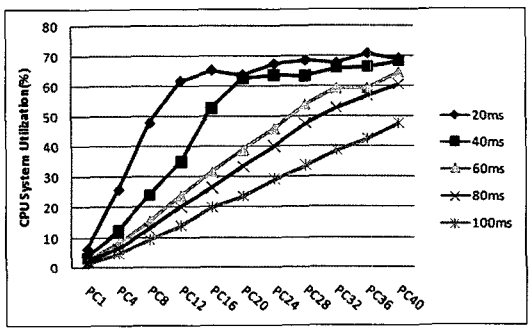

그림 4 Host1과 Host3 간에 CPU 시스템 사용률

3.2.1.3 Host1과 Host4의 경우

표 6과 7은 Host1과 Host4 간의 데이타 전송 시 측정된 전송률과 CPU 시스템 사용률을 나타내고 있다. 이번 실험에서도 CPU 사용률과 전송률이 비례함을 알 수 있다. 그러나 이전의 실험과 다른 점은 RTT가 증가될 때 병렬연결의 개수를 증가시켜도 CPU 시스템 사용률이 점차 감소되는 것이다. 이는 호스트 성능이 매우 낮아 증가된 병렬연결들을 원활히 수행할 수 없기 때문

표 6 Host1과 Host4 간의 전송률 (단위: Mbps)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	23.85	11.97	8.03	6.03	4.83
PC_4	94.21	47.77	32.04	24.1	19.3
PC_8	133.15	95.48	63.96	48.09	38.22
PC_{12}	160.09	110.26	84.77	69.10	57.39
PC_{16}	165.37	128.37	92.51	81.69	69.03
PC_{20}	196.6	145.16	116.52	94.99	80.75
PC_{24}	209.75	157.53	127.75	107.59	91.98
PC_{28}	198.85	143.37	135.14	116.54	90.65
PC_{32}	194.87	144.45	133.1	110.05	110.09
PC_{36}	187.27	139.56	115.7	97.91	92.02
PC_{40}	195.01	142.51	114.65	96.53	86.96

표 7 Host1과 Host4 간의 CPU시스템사용률 (단위: %)

RTT / PC_i	20ms	40ms	60ms	80ms	100ms
PC_1	9.09	5.23	3.11	2.35	1.85
PC_4	36.7	18.53	12.18	9.59	7.66
PC_8	51.66	37.11	24.76	19.69	15.35
PC_{12}	62.75	43.36	32.94	28.64	22.86
PC_{16}	62.78	50.86	36.67	32.56	26.77
PC_{20}	76.27	57.35	46.84	38.32	32.35
PC_{24}	81.26	61.48	49.95	43.15	36.28
PC_{28}	81.26	57.28	52.22	45.54	35.94
PC_{32}	74.55	58.66	51.03	42.1	42.89
PC_{36}	72.44	55.48	44.33	38.83	36.87
PC_{40}	75.66	56.49	44.7	37.86	34.87

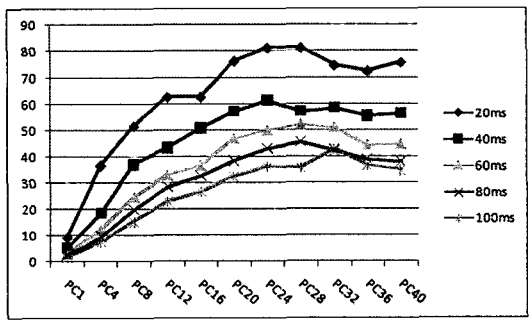

그림 5 Host1과 Host4 간에 CPU 시스템 사용률

으로 고려해 볼 수 있다.

그림 5는 위의 표 7을 그래프로 나타낸 것이다.

Host2와 Host3의 실험결과에서 본 것처럼 Host4에서 역시 비슷한 형태의 CPU 시스템 사용률의 그래프를 보여주고 있다. 이번 실험 결과에서도 마찬가지로 RTT가 증가할수록 그래프의 기울기가 낮아지는 현상을 볼 수 있다.

3.2.2 상관관계 분석

표 8은 위의 실험에서 얻은 데이타를 이용한 것으로

표 8 병렬연결수와 CPU 시스템 사용률의 상관관계계수

Host / RTT	Host 2	Host 3	Host 4
20ms	0.652376	0.805649	0.804341
40ms	0.801089	0.917061	0.824442
60ms	0.886845	0.990332	0.846808
80ms	0.954631	0.997032	0.847921
100ms	0.990719	0.99963	0.904319

각각의 RTT에서 병렬연결수와 CPU 시스템 사용률간의 상관관계 계수를 나타낸 것이다.

RTT가 짧을수록 상관관계 계수가 낮아지는 현상을 볼 수 있는데, 이는 적은 수의 병렬연결 개수로도 최대 CPU 시스템 사용률을 획득할 수 있으므로 나타나는 현상이라 볼 수 있다. 표 3을 보면, RTT 20ms인 환경에서 PC_8을 이용할 경우 CPU 시스템 사용률은 78.64%로 측정되었으나 TCP 연결 개수가 증가함에도 불구하고 더 이상의 증가는 없었다. 따라서 RTT가 짧은 경우 상관관계계수는 낮게 나타난다. 또한 RTT가 길어질수록 상관계수가 증가하며, 결국, RTT 100ms에서는 세 호스트 모두 상관관계 계수가 0.9 이상으로 매우 밀접한 관계를 나타냈다. 이는 RTT가 길어질수록 더 많은 TCP 연결이 요구되기 때문이다.

3.2.3 CPU 시스템 사용률의 추이 분석

3.2.3.1 RTT 증가에 따른 추이분석

표 9는 동일한 병렬연결 개수에서 RTT가 증가함에 따라 나타나는 CPU 시스템 사용률의 변화를 추세식으로 나타내고 있다.

각 식에서 지수승들은 RTT의 증가에 따라 감소되는 CPU 시스템 사용률의 비율을 의미하는데, 이 지수승들의 오차를 평균 계산하여 '지수승 평균 오차' 열에 나타냈다. 이 표를 통해 PC_1과 PC_{12}의 지수승 평균 오차가 0.05로, 세 호스트의 CPU 시스템 사용률 감소 비율이 가장 유사함을 확인할 수 있다. 즉, 이것은 한 RTT에서 PC_1 시 사용되는 CPU 시스템 사용률을 측정하였을 경우, 다른 RTT에서 PC_1 사용 시 이용될 CPU 시스템 활용률을 예측할 수 있음을 의미한다.

3.2.3.2 병렬연결 개수 증가에 따른 추이 분석

측정된 데이타를 분석한 결과 CPU 시스템 사용률이 병렬연결 개수, 전송률, RTT와 밀접한 관계가 있음을 알 수 있었다. 따라서 현재 측정되는 CPU 시스템 사용률을 이용한다면 최대 전송 성능을 낼 수 있는 병렬연결의 개수를 예측할 수 있을 것으로 사료된다. 이에 CPU 시스템 사용률의 병렬연결 개수에 따른 변화를 분석하고자 한다. 수집된 데이타를 보면 병렬연결 개수와 CPU 시스템 사용률의 증가는 어느 정도 비례함을 알 수 있었는데, 이번 단락에서 이를 확인하고자 한다.

표 9 병렬연결 수에 따른 CPU 시스템 사용률 추세식

PC_i \ 호스트명	Host2	Host3	Host4	지수승 평균 오차
PC_1	$165.6 \times RTT^{-0.9539}$	$180.94 \times RTT^{-0.9893}$	$92.415 \times RTT^{-0.9003}$	0.059
PC_4	$708.2 \times RTT^{-0.975}$	$587.06 \times RTT^{-1.037}$	$1052.7 \times RTT^{-1.2994}$	0.216
PC_8	$1412.8 \times RTT^{-0.9761}$	$880.8 \times RTT^{-0.9688}$	$1863.7 \times RTT^{-1.2123}$	0.162
PC_{12}	$1204.4 \times RTT^{-0.8423}$	$1009.6 \times RTT^{-0.908}$	$910.42 \times RTT^{-0.9204}$	0.052
PC_{16}	$796.4 \times RTT^{-0.6781}$	$883.51 \times RTT^{-0.8062}$	$507.8 \times RTT^{-0.7094}$	0.085
PC_{20}	$282.1 \times RTT^{-0.3852}$	$600.64 \times RTT^{-0.6639}$	$335.26 \times RTT^{-0.5643}$	0.186
PC_{24}	$245.1 \times RTT^{-0.3199}$	$396.06 \times RTT^{-0.5324}$	$169.14 \times RTT^{-0.3673}$	0.142
PC_{28}	$127.3 \times RTT^{-0.1438}$	$287.41 \times RTT^{-0.428}$	$102.46 \times RTT^{-0.2254}$	0.19
PC_{32}	$88.6 \times RTT^{-0.051}$	$218.19 \times RTT^{-0.3416}$	$53.564 \times RTT^{-0.055}$	0.194
PC_{36}	$66.4 \times RTT^{0.02}$	$185.76 \times RTT^{-0.2871}$	$49.054 \times RTT^{-0.0167}$	0.191
PC_{40}	$119.4 \times RTT^{-0.1295}$	$135.65 \times RTT^{-0.1987}$	$46.81 \times RTT^{0.0026}$	0.133

다음 (1)과 같은 식을 이용하여 CPU 시스템 사용률을 예측할 수 있으며, 표 10은 이 예측값과 실제 측정된 CPU 시스템 사용률을 나타내고 있으며, 그 오차 또한 함께 보여주고 있다. 식 (1)에서 CU_{rtt}는 RTT에서 PC_1 사용 시 측정된 CPU 시스템 사용률을 의미하며, PCU는 PC_i를 이용할 경우 CPU 시스템 사용률의 예측한 값이다. 단, CPU 시스템 사용률은 100이하의 값을 가지므로 PCU가 100이상일 경우는 99로 표시하였다.

$$PCU = CU_{rtt} \times PC_i \quad (1 \le i \le 40,\ 20 \le rtt \le 100) \quad (1)$$

표 10에서 전송 호스트와 수신 호스트 간의 RTT가 60ms인 경우, PC_1에서 CPU 시스템 사용률이 3.33%로 측정되었다. 식 (1)에 의하면 PC_4를 사용할 경우 PCU는 13.34%로 계산되며 실제 측정된 CPU 시스템 사용률은 13.33%이다. 따라서 오차는 0.01%이다. 또한 PC_8을 사용할 경우에는 예측되는 값이 26.67%이며, 실제 측정값은 25.90%로 오차는 0.77%로 나타났다.

RTT 20ms, 40ms, 80ms, 100ms 등의 다른 RTT의 경우에서도 이와 같은 현상이 나타난다. 그러나 측정된 CPU 시스템 사용률이 약 70%이상인 경우 이들 CPU 시스템 사용률의 실제값과 PCU는 오차가 발생된다. 이는 CPU 시스템 활용에 있어 부하가 발생되기 때문으로 볼 수 있다. 그러므로 최대로 활용할 수 있는 CPU 시스템 사용률까지는 병렬연결 개수와 CPU 시스템 사용률은 정비례 관계가 형성되어 있음을 알 수 있다. Host3과 Host4의 경우에도 같은 결과를 얻을 수 있었다.

3.2.3.3 최대 CPU 시스템 사용률 추이 분석

그림 6은 Host2, 3, 4에서 RTT 별로 측정된 최대 CPU 사용률을 나타낸 그래프이다. 여기서 사용된 최대 CPU 시스템 사용률은 실제 측정값과 PCU 간의 오차가 5이상 벌어지는 구간의 CPU 시스템 사용률들을 평균한 값이다.

표 10 PCU와 측정값의 비교 (단위 : %)

PCi \ RTT		20ms	40ms	60ms	80ms	100ms
PC1	측정	9.41	4.93	3.33	2.54	2.05
	PCU	9.41	4.93	3.33	2.54	2.05
	오차	0.00	0.00	0.00	0.00	0.00
PC4	측정	37.61	19.34	13.33	10.06	7.89
	PCU	37.63	19.71	13.34	10.15	8.19
	오차	0.01	0.37	0.01	0.08	0.31
PC8	측정	78.64	38.72	25.90	19.79	15.78
	PCU	75.25	39.42	26.67	20.29	16.38
	오차	3.38	0.70	0.77	0.50	0.60
PC12	측정	73.69	58.51	38.05	29.33	23.61
	PCU	99.00	59.13	40.01	30.44	24.57
	오차	25.31	0.62	1.96	1.11	0.97
PC16	측정	82.20	86.03	53.22	39.54	31.24
	PCU	99.00	78.84	53.34	40.58	32.77
	오차	16.80	7.19	0.12	1.04	1.53
PC20	측정	62.37	79.30	66.79	48.78	39.29
	PCU	99.00	98.55	66.68	50.73	40.96
	오차	36.63	19.25	0.11	1.95	1.67
PC24	측정	84.56	74.64	79.19	59.42	46.31
	PCU	99.00	99.00	80.02	60.87	49.15
	오차	14.44	24.36	0.83	1.46	2.84
PC28	측정	71.60	76.97	74.59	71.55	54.23
	PCU	99.00	99.00	93.35	71.02	57.34
	오차	27.40	22.03	18.76	0.53	3.12
PC32	측정	81.31	68.86	73.03	70.00	64.93
	PCU	99.00	99.00	99.00	81.17	65.53
	오차	17.69	30.14	25.98	11.17	0.60
PC86	측정	77.84	79.16	69.14	76.68	78.03
	PCU	99.00	99.00	99.00	91.31	73.72
	오차	21.16	19.84	29.86	14.64	4.30
PC40	측정	78.22	81.47	69.69	64.47	70.56
	PCU	99.00	99.00	99.00	99.00	81.92
	오차	20.78	17.53	29.31	34.53	11.36

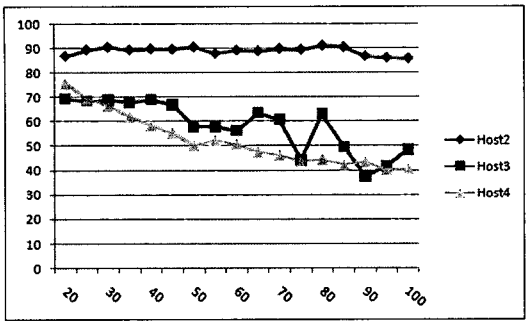

그림 6 RTT 증가에 따른 최대 CPU 시스템 사용률

그림 6을 보면, 낮은 성능의 호스트일수록 전송 거리가 멀어질수록 최대 CPU 시스템 사용률은 감소하는 현상을 볼 수 있다. 이는 호스트의 CPU 성능이 많은 수의 TCP 연결을 운영하는데 역부족하기 때문으로 볼 수 있다. 따라서 Host2에서는 최대 CPU 시스템 사용률의 감소는 거의 보이지 않으며, Host3과 Host4에서는 최대 CPU 시스템 사용률이 감소되고 있다. 또한 Host4는 Host3보다 감소되는 비율이 높다.

그러나 본 연구에서는 최대 CPU 시스템 사용률의 감소비율을 적용하지 않고자 한다. 앞으로 호스트 CPU의 성능은 계속 향상될 것이며, 낮은 성능의 호스트에서 최대 CPU 시스템 활용률을 높게 설정함으로써 전송 능력을 끌어올리기 위함이다. 따라서 본 연구에서는 단거리, 즉, 20ms 이하에서 측정된 최대 CPU 시스템 사용률을 모든 RTT에서 얻을 수 있는 최대 CPU 시스템 사용률로 적용한다.

4. 병렬연결 개수 예측 모델링

서로 다른 성능의 호스트임에도 불구하고 PC_1을 사용할 경우 RTT의 변화에 따른 CPU 시스템 사용률의 변화율이 유사함을 위의 3절에서 확인하였다. 본 논문에서는 이러한 특성을 이용하여 최대 성능을 발휘할 수 있는 병렬연결 개수를 예측한다. 즉, 어느 한 RTT에서 PC_1로 CPU 시스템 사용률을 측정할 경우, 다른 RTT에서 PC_1 사용할 수 있는 CPU 시스템 사용률을 예측할 수 있다. 예측을 원하는 RTT에서 PC_1을 이용하여 데이타를 전송할 경우 사용될 CPU 시스템 활용률(CU_{RTT})은 다음 식 (2)를 이용하여 계산할 수 있다. e는 -0.94783으로 표 9에서 PC_1을 이용하는 세 호스트의 추세식 지수, -0.9539, -0.9893, -0.9003을 평균 계산한 값이다.

$$CU_{RTT} = S \times RTT^{e} \quad (2)$$

S는 호스트의 성능에 따라 다른 값을 가지며, 식 (3)과 같이 계산할 수 있다. 임의의 rtt에서 PC_1로 측정된 CPU 시스템 사용률을 이용하여 계산된다.

$$S = \frac{CU_{rtt}}{rtt^{e}} \quad ③$$

최대 CPU 시스템 사용률(MCU)은 RTT 20ms 이하의 단거리에서 다수의 TCP 연결(PC_{20} 이상)을 이용하여 실제 데이타를 전송할 경우 얻을 수 있다. 따라서 최대 전송 성능을 유도할 수 있는 병렬연결 개수(PCN)는 다음 식 (4)로 얻을 수 있다.

$$PCN = \frac{MCU}{CU_{RTT}} \quad (4)$$

5. 성능 평가

5.1 실험 환경

4장에서 제시된 내용을 본 장에서는 두 대의 호스트를 이용하여 검증해 보고자 한다. 실험 환경은 다음 그림 7과 같다. 3장에서의 실험 환경과 같은 1GMbps의 스위치허브에 호스트들이 연결되어 있으며, 1Gbytes의 데이타가 Host5에서 Host1로, Host6에서 Host1로 전송된다. 검증을 위해 사용된 호스트의 성능은 다음 표 11과 같다. Host1은 3장에서와 같이 Nistnet을 수행하여 RTT를 조절하며, 추가적으로 Packet loss 비율도 조절한다.

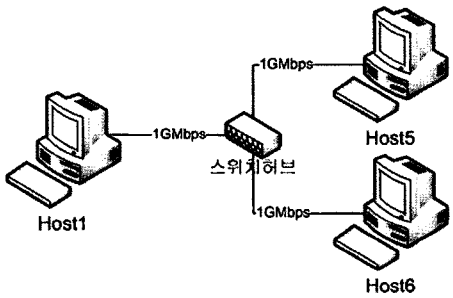

그림 7 검증을 위한 실험 환경

5.2 실험 결과

5.2.1 Host5에 적용한 결과

이번 실험은 Packet loss가 5%인 경우 RTT 30ms에서 PC_1로 CPU 시스템 사용률을 측정한 경우 수행된 실험이다. CU_{30}은 4.35로, 목표 MCU는 61.13로 측정되었다. 따라서 S는 109.17로 계산되며, 이를 이용해 계산된 예측 CU_{RTT}는 표 12에 나타나 있다. 또한 측정된 CU_{RTT}도 표 12에 나타나 있어 예측 CU_{RTT}와 측정된 CU_{RTT}간의 오차도 함께 볼 수 있다. 표 12의 오차를 보면 예측값과 실제값이 거의 일치함을 알 수 있다.

각 RTT에서의 예측된 CU_{RTT}와 목표 MCU를 이용하여 PCN을 계산하고 그 결과를 표 13에 나타냈다. 또

표 11 검증 호스트의 성능

구분 / 호스트명	CPU	Memory	LAN card	OS
Host1	Intel Celeron CPU 2.8GHz (cpuMHz 2795.256)	482180KB	32bit GIGA bit PCI LanCard GIGA-1000s	Fedora4
Host5	Intel Pentium 4 1.6GHz (cpuMHz 1594.860)	254756KB	32bit GIGA bit PCI LanCard GIGA-1000s	RedHat9 2.4.20-8
Host6	Intel Celeron CPU 2.8GHz (cpuMHz 2795.241)	482180KB	Intel 8390MT 32bit UTP	RedHat9 2.4.20-8

표 12 CU_{RTT} 예측값과 실제값의 비교

구분 / RTT	예 측	측 정	오 차
25ms	5.17	5.27	0.11
35ms	3.75	3.78	0.02
45ms	2.96	2.93	0.02
55ms	2.45	2.71	0.26
65ms	2.09	2.22	0.13
75ms	1.82	1.97	0.14
85ms	1.62	1.82	0.20
95ms	1.46	1.45	0.01

표 13 *PCN*과 *PCN* 이용 시 전송률과 *MCU*

구분 / RTT	PCN	전송률	목표 MCU	측정 MCU	오 차
25ms	12	202.14	61.83	59.05	2.77
35ms	16	193.52	61.83	56.53	5.30
45ms	21	196.40	61.83	58.84	2.98
55ms	25	169.98	61.83	58.24	3.59
65ms	30	200.47	61.83	58.32	3.51
75ms	34	192.03	61.83	58.48	3.34
85ms	38	191.20	61.83	57.12	4.71
95ms	42	192.36	61.83	57.33	4.49

한 계산된 PCN을 이용하여 실제 데이타 전송을 수행하고 측정된 MCU와 목표 MCU를 비교하고 있다.

두 MCU 간의 평균 오차가 3.27로 나타나 계산된 PCN을 사용할 경우 최대 전송 성능에 근접함을 알 수 있다.

5.2.2 Host6에 적용한 결과

Packet loss가 5%인 경우 RTT 90ms에서 PC_I로 CPU 시스템 사용률을 측정한 경우 수행된 실험이다. CU_{90}은 0.868로 측정되었고, 목표 MCU는 27.40으로 측정되었다. CU_{90}을 이용하여 계산되는 S는 61.79이다. 표 14에는 각 RTT에서 계산된 CU_{RTT}와 실제 측정된 CU_{RTT}가 비교되어 있으며 평균 오차는 0.05로 나타났다. 그러므로 실제 CU_{RTT}와 예측 CU_{RTT}는 매우 근사함을 알 수 있다.

표 14에는 계산된 PCN과 이를 이용하여 실제 데이타를 전송하고 측정된 MCU와 목표 MCU를 비교하고 있다.

표 14 CU_{RTT} 예측값과 실제값의 비교

구분 / RTT	예 측	측 정	오 차
25ms	2.92	2.90	0.02
35ms	2.46	2.32	0.14
45ms	1.87	1.77	0.10
55ms	1.52	1.49	0.02
65ms	1.28	1.27	0.01
75ms	1.10	1.06	0.04
85ms	0.97	0.93	0.04
95ms	0.82	0.83	0.00

표 15 *PCN*과 *PCN* 이용 시 전송률과 *MCU*

구분 / RTT	PCN	전송률	목표 MCU	측정 MCU	오 차
25ms	9	166.39	27.40	22.19	5.21
35ms	11	169.62	27.40	22.38	5.02
45ms	15	169.96	27.40	24.05	3.35
55ms	18	162.39	27.40	22.52	4.88
65ms	21	172.34	27.40	23.33	4.07
75ms	25	147.72	27.40	22.81	4.59
85ms	28	158.29	27.40	22.10	5.31
95ms	33	158.86	27.40	22.76	4.64

Host6에서 사용한 Lan card는 인터럽트 완화기능을 가지고 있어 CPU 점유율을 크게 낮추어 처리 능력 향상시키는 기능을 가지고 있다. 따라서 목표 MCU가 Host5와 다르게 매우 낮은 현상이 나타난다. 이렇게 특수한 Lan card를 이용할 경우에 실험 결과는 측정된 MCU과 예측된 MCU 간의 평균 오차가 4.52로 Host5의 경우에 비해 다소 오차가 나는 것으로 나타났다. 그러나 근소한 차이이므로 최대 성능에 근접하였다고 볼 수 있다.

5.2.3 성능 평가 결과

표 16은 위의 실험에서 CU_{RTT}의 실제값과 예측값의 오차를 나타내고 있으며, 목표 MCU와 PCN 이용 시 측정된 MCU 간의 오차를 나타내고 있다. CU_{RTT} 오차가 작다는 것은 임의의 RTT에서 사용될 CPU 시스템 사용률을 정확하게 예측하고 있다는 의미가 되며, 이는

표 16 호스트 별 CU_{RTT}, MCU 평균 오차 비교

구분 / 호스트명 \ packet loss		CU_{RTT} 오차	MCU 오차
Host5	0%	0.19	3.48
	5%	0.25	2.74
Host6	0%	0.10	4.78
	5%	0.05	3.86

곧 최대 전송 성능을 획득할 수 있는 다중 TCP 연결 개수를 보다 정확하게 계산 가능함을 의미한다. 또한 MCU 오차가 작다는 것은 계산된 PCN을 이용할 경우 목표 MCU에 근접함을 의미하는 것으로 본 논문에서 제안한 메커니즘의 정확성을 보여준다.

위의 실험은 packet loss가 발생되는 환경과 그렇지 않은 환경에서 수행하여 packet loss가 PCN의 계산에 영향을 미치는지 알아보았고, ,또한 다른 종류의 Lan card를 사용하여 실험을 함으로써 PCN 예측이 Lan card의 영향을 받는지를 확인해 보았다. 표 12~15에서 볼 수 있듯이 packet loss와 Lan card는 PCN 결정에 큰 영향을 미치지 않는 것으로 나타났다.

6. 결 론

본 논문에서는 다중 TCP 연결과 호스트 성능과의 관계를 분석하고 이를 이용하여 데이타 전송에서 최대 성능을 획득할 수 있는 병렬연결 개수 결정 메커니즘을 연구하였다. 수집된 데이타 분석을 통하여 CPU 시스템 사용량과 병렬연결 개수와의 밀접한 관계가 있음을 밝혔고, 이를 이용해 최대 활용 가능한 CPU 시스템 사용량을 예측함으로써 데이타 전송 성능을 극대화하였다.

그러나 본 연구에서는 트래픽이 많이 존재하는 네트워크 상황에 대해서는 고려하지 않았다. 병렬 전송 기술은 TCP의 문제로 인해 고성능 네트워크를 제대로 활용할 수 없기에 개발된 것으로 네트워크에 부하가 없는 상황에 이용될 수 있는 기술이다. 즉, 네트워크에 트래픽이 많을 경우 병렬 전송 기술을 사용하게 되면 TCP 연결들 간의 간섭현상이 발생하며, 또한 네트워크 혼잡 등의 다른 문제도 발생하게 된다. 따라서 본 논문에서는 트래픽에 관련된 연구는 향후로 미루고 다루지 않았다.

그러나 이러한 간섭 현상에 대한 연구는 특정 기관 간의 다중 TCP 연결 관리 시스템을 개발할 수 있는 기반이 될 수 있으며, 이러한 시스템에서는 TCP 연결의 개수를 이용하여 QoS 등도 지원이 가능할 수 있을 것으로 생각된다.

참 고 문 헌

[1] T. Dunigan, M. Mathis and B. Tierney, "A TCP Tuning Daemon," Proceeding of IEEE Supercomputing 2002 Conference, Baltimore, Maryland, LBNL-51022, Nov. 2002.

[2] Yunhong Gu, Xinwei Hong, Marco Mazzucco, Robert Grossman, "SABUL: A High Performance Data Transfer Protocol," IEEE COMMUNICATIONS LETTERS, 2003.

[3] Aaron falk, Ted Faber, Joseph Bannister, Andrew Chien, Robert Grossman, Jason Leigh, "Transport Protocols for High Performance," Communications of The ACM, Vol.46, No11, November 2003.

[4] Eric Weigle, Wu-chun Feng, "Dynamic Right-Sizing: A Simulation Study," IEEE ICCCN, 2001.

[5] J.Semke, J.Mahdavi, M.Mathis, Automatic TCP Buffer Tuning, ACM SIGCOMM 1998, Vol.28, No.4, 1998.

[6] Gi-chul Yoo, Eun-sook Sim, Dongkyun Kim, Taeyoung Byun, Kookhan Kim, Okh- wan Byun, "An Efficient TCP Buffer Tuning Technique Based on Packet Loss Ratio (TBT-PLR)," International Conference on Internet Computing, 2004.

[7] White paper GridFTP Universal Data Transfer for the Grid, 2000.

[8] Harimath Sivakumar, Stuart Bailey, Robert L.Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks," SC2000: High-Performance Network and Computing, 2000.

[9] Thomas J. Hacker, Brain D. Athey, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network," Parallel and Distributed Processing Symposium (IPDPS), 2002.

[10] Bill Allcock, Igor Mandrichenko, Timur Perelmutov, "GridFTP v2 Protocol Description," Germi National Accelerator Laboratory, 2004.

[11] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, "GridFTP: Protocol Extensions to FTP for the Grid," Argonne National Lab., April 2002.

[12] Pawel Plaszczak, Joe Link, Rich Wellner, "GridFTPexplained," Paul Hubbard, http://www.icslab.agh.edu.pl/~kzajac/pa-wel/GridFTPexplained.doc

[13] Kun Myon Choi, Eui-Nam Huh, and Hyunseung Choo, "Efficient Resource Management for TCP Parallel Streams with Buffer Tuning," Springer-Verlag Lecture Notes in Computer Science, Vol. 3823, pp. 683-692, December 2005.

[14] "NISTnet A Linux based Network Emulation Tool," http://snad.ncsl.nist.gov/nistnet/, retrieved on June 2003.

[15] Young-Sin Kim, Eui-Nam Huh, "Analysis of the Interference between Parallel Sockets Connections and Predictions of the Bandwidth," International Journal of Computer Science and Network Security, 2005.

김 영 신

1999년 서울여자대학교 컴퓨터학과 졸업(학사). 2002년 서울여자대학교 대학원 컴퓨터학과 졸업(석사). 2007년 서울여자대학교 대학원 컴퓨터학과 졸업(박사) 2007년~ 경희대학교 전자정보대학 연구박사. 관심분야는 센서네트워크, 보안

허 의 남

1990년 부산대학교 전산통계학과 졸업(학사). 1995년 Univ. of Texas at Arlington 컴퓨터공학과 졸업(석사). 2000년 Ohio University 컴퓨터공학과 졸업(박사). 2002년 삼육대학교. 2003년 서울여자대학교 정보통신공학부 조교수. 2005년~현재 경희대학교 전자정보공학부 부교수. 관심분야는 텔레메틱스, 그리드 컴퓨팅, 센서 네트워크, 보안