論 文

종합정보망(INS)을 위한 command/file server의 연구

正會員 李 哲 洙* 正會員 李 華 淵** 正會員 黃 文 俊***

A Study on the Command/File Server for INS

Chul Soo LEE*, Hwa Yeon LEE**, Moon Joon HWANG*** *Regular Members*

要 約 86아시안 게임에 사용되었던 종합정보망(INS) system은 Ethernet-based LAN을 통하여 6대의 supermini-computer를 연결하여 분산 처리를 하였다. 이를 위해 UNIX OS상에서 UTP라는 command/file server를 개발하여 분산시스템 간에 data consistency와 user transparency를 가능하게 하였다. 게임 기간 동안 UTP는 평균적으로 780byte/sec의 전송률을 보였다.

ABSTRACT Data, distributed and/or duplicated among multiple computers interconnected by a LAN, need to be managed consistently and accessed transparently by users. A command/file server, named UTP, which meets the preceding needs was developed for supermini-computers which are interconnected by an Ethernet-based LAN and run under UNIX OS.

I. 서 론

1970년대를 data base의 기간이라고 한다면, 1980년대는 분산 처리의 기간이라고 할 수 있다. 컴퓨터와 통신 기술의 결합으로 컴퓨터 시스템의 구성과 이용에 큰 변화가 왔다. 최근에는 한 개의 대형 컴퓨터로 모든 일을 처리하는 방식에서 여러곳에 분산되어 있는 컴퓨터를 서로 연결하고 기능을 분산하는 방식으로 바뀌어 가고 있다. 이렇게 기능을 분산시킴으로써 각 컴퓨터등은 각각 화일 관리, 터미날 처리 등 특정 기능을 수행하도록 하여 전체 시스템의 구성을 쉽고 효과적으로 할 수 있게 되었다.

86 아시안 게임에 사용되었던 종합정보망(INS) system은 〈그림 1〉과 같이 6대의 3B20S (supermini computer)와 2대의 3B2 (micro-mini computer)를 LAN(3BNET)으로 연결하였다.

3B 20S는 다시 4대의 service(SVC)node와 1대의 monitoring(MM) node, 1대의 external interface(EI) node로 구분한다. service node는 일반 user에게 경기 결과를 보여주거나,

*, **, ***韓國데이타通信(株)
Data Communications Corp. of Korea
論文番號 : 87-44(接受 1987. 3. 6)

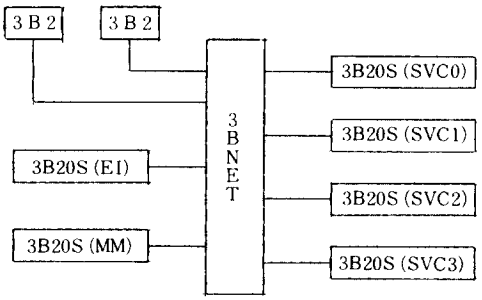


그림 1 INS 시스템 구성
INS system configurtion

electronic-mail 기능을 제공하는 일을 주로 담당한다. MM node는 INS system 전반에 걸쳐 system load를 monitor하고 각종 통계를 작성한다. EI node는 경기 결과를 각 service node로 전달하는 역할을 한다.

공지사항, 게시판, 경기 결과 uid 등의 data는 모든 node가 가지고 있다. 만일 어느 한 node에서 data를 수정하더라도 모든 node에서 같이 수정되어야 한다. 한편 각 개인의 mail-box는 uiddb에 등록된 대로 어느 한 node에 있지만 user는 mail-box의 위치에 관계없이 어느 node 에서도 mail을 보내기도 하고 읽어볼 수도 있어야 한다. INS에서는 이와 같이 각 node에 분산되어 있는 각종 data들의 consistency를 유지하면서 user에게는 node에 관계없이 각종 data를 사용할 수 있도록 user transparency를 제공하기 위해, UTP(User Transparency Provider) 라는 network software를 개발하였다. UTP는 화일 전송 뿐만 아니라, RCE(Remote Commard Execution) 기능도 가지고 있다. 〈그림 2〉에 UTP 관련 communication architecture를 보였다. 먼저 3BNET(NIB, ni)에 대해 설명하고 나서 UTP를 논하기로 한다.

〔주〕 uid는 electronic-mail등과 같은 특수 service를 사용할 수 있는 사람들에게 부여하는 고유 번호로 uiddb에는 각 개인의 이름, 국적, mbox의 위치, 각 service별 permission 등이 들어 있다.

Ⅱ. 3BNET

가. 3BNET의 개요

3BNET은 UNIX를 사용하는 host computer들과 기타 주변 장치들을 상호 연결하는 Ethernet-based LAN으로, 전송매체는 동축 케이블이며 전송속도는 10Mbps이다. 3BNET에 연결된 computer나 주변 장치를 node라고 하며 각

4	User	User Application	경기 결과, 공지 사항, 전자 우편, 기타
3	UTP	End-to-End	Command/File server
2	ni	UNIX kernel	system call 처리
1	NIB	3BNET Protocol	cyclic redundancy check
		Ethernet function	data encapsulation/decapsulation link management collision handling
			encode/decode transmit/receive collision detection carrier sensing

그림 2 UTP 통신 계층 구조
Communication Architecture of UTP

node를 master, backup, slave node로 구분한다. master node는 network의 상태나 traffic의 기록, configuration의 관리등을 담당한다. 즉, master node는 network의 상태와 configuration에 대한 정보를 가지고 있으며, 주기적으로 network의 integrity를 조사한다. 만일 master node가 down되거나 master node를 변경하면 backup node가 master node로 된다.

data의 전송은 CSMA/CD방식에서 packet단위로 이루어지며 전송 protocol로서 Ethernet protocol과 3BNET protocol의 두가지 mode가 있는데, 후자의 경우 CRC(Cyclic Redundancy Check) 기능이 있어서 channel상의 bit error가 발생할 경우 수신 packet을 무시한다. 최대 packet size는 4K byte이다.

나. NIB와 ni의 개요

3BNET의 NIB(network interface board)는 Ethernet의 controller와 비슷한 기능을 가지고 있다. NIB는 cpu, buffer, interface의 3 board로 구성되며 DMA를 통해 host상의 main memory와 data transfer가 일어난다.

NIB와 user process 사이에서 data는 NIB buffer, ni system buffer, user buffer의 순서로 전달되지만, shared memory를 사용하면 ni system buffer를 거치지 않고 직접 NIB buffer에서 shared memory로 옮겨지므로 memory copy 회수가 줄어든다.

ni는 UNIX의 관점에서 볼 때 하나의 I/O driver이다. 따라서 ni는 NIB를 통해 data를 송수신하고 결과를 user process에게 전달한다. user process가 ni의 service를 받기 위해 UNIX system call인 open( ), close( ), read( ), write( ) 그리고 ioctl( )이 제공되고 있다.

ni는 64개의 local port를 가지고 있는데, 새로운 port의 할당 및 회수, data의 송수신을 위하여 NIB와 ni는 각 port별로 control-table, response-queue, receive-buffer(ni system buffer)를 가지고 있다. 〈그림 3〉에 ni 와 NIB의 관계를 나타냈다.

각 node마다 ntq[ ](network tansmit queue)가 있는데, NIB에게 command를 전달하고 싶은 ni는 commad를 작성하여 ntq[ ]에 넣어 두고 자신은 sleep( )하고 있으면, NIB가 ntq[ ]에서 command를 꺼내 수행하고 그 결과를 해당 port의 response-queue에 써 넣고 sleep( )중인 process를 wakeuq( ) 시킨다. data를 read( ) 하려는 경우에는 수신한 data는 receive-buffer 또는 shared memory에 들어 간다.

〔주〕 앞으로 [ ]는 array를 의미한다.

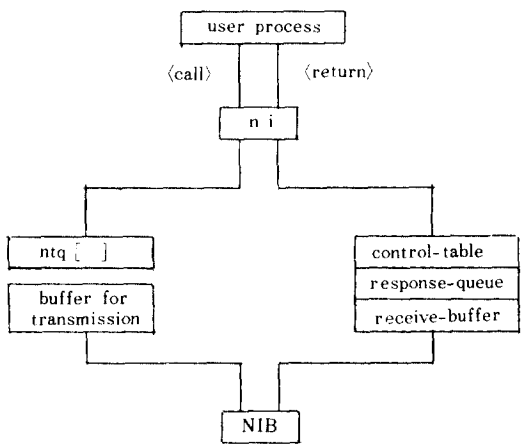


그림 3 NIB와 ni의 관계
Interrelations between NIB and ni

다. Programming interface to ni

user는 NIX가 제공하는 system call인 open( ), close( ), read( ), write( ), ioctl( )에 의해 ni의 service를 받을 수 있다. UTP는 이 system call을 이용하여 구현하였다. 각 node마다 64개의 local port가 있는데 open( ), close( )를 통해 이 port를 할당받고, 반납한다. channel setup 요구와 data(file, command)는 read( ), write( )에 의해 packet형태로 교환하였다. address 지정에는 ioctl( )을 이용하였다.

각 system call에 대해 살펴 본다.

• open( )

node마다 64개의 port가 있는데, ni를 통해 communication을 하기 위해서는 open을 통해 port를 하나 할당 받아야 한다. 형식은 다음과 같다.

fd = open("/dev/ni", oflag) ;

oflag는 O-RDONLY, O-WRONLY, O-RDWR중 하나가 된다. 이것은 각각 receive-only, transmit-only, full-duplex를 의미한다. oflag의 O-NDELAY bit를 0로 하면 read( ) 하려고 하는 경우 data가 아직 수신되지 않았으면 수신될 때까지 기다린다.

• close( )

close는 할당되었던 port를 반납하는 것이다. 형식은 다음과 같다.

close(fd) ;

close에서는 먼저 ntq[ ]를 통해 CLOSEP라는 command를 NIB에게 넘겨 NIB에서 내부적으로 close와 관련된 작업이 이루어지게 한다. 그리고 port에게 할당되었던 buffer 들을 반납한다. 그리고 control-table에 port가 반납되었음을 기록하여 새로 open( ) 하는 process에게 이 port가 할당될 수 있도록 한다.

• ioctl ( )

ni와 NIB는 control-table을 통해 서로 interface가 이루어지며 ioctl( )에 의해 각 항목을 변경한다. 그 형식은 다음과 같다.

```
ioctl(fd, NIGETA, &lpstatus)
/* read table */
ioctl(fd, NISETA, & lpstatus);
struct lpstatus {
        short rcvb-sz ;
/* receive buffer size */
short rcvq-sz ;
/* receive queue size */
short rspq-sz ;
/* response queue size */
long rcvb-loc ;
/* receive buffer location */
long xmitb-loc ;
/*transmit buffer location */
char netid[8] ;
/* my netid(address) */
char tonid[8] ;
/*destination netid (address) */
```

open( )을 하면 control-table의 일부만 default로 set되므로, 일반적으로 open( ) 하고는 ioctl (NIGETA)를 하여 table을 읽고, 일부 값을 원하는대로 변경하여 ioctl (NISETA)를 통해 다시 써 넣는다.

lpstatus의 모든 항목이 임의로 변경될 수있는 것은 아니다. 송수신을 위한 queue나 buffer의 위치와 size는 한번 결정되면 close( ) 하기 전에는 변경될 수 없다. 자신의 address도 한번 정해지면 변경할 수 없다. 그러나 상대방의 address는 수시로 변경 가능하다. 첫 ioctl(NISETA)때 ntq[ ]를 통해 OPENP라는 command를 NIB에게 전달하여 NIB로 하여금 새 port에 대한 data 송수신이 가능하도록 한다.

• read( )

data의 수신에 쓰인다. 형식은 다음과 같다.

read-count=read(fd, buffer, size) ;

ni는수신된 data가 없다면 기다리지 않고 return한다. 그러나 open( )할 때 oflag에 O-NDELAY bit가 0로 set되어 있었다면 data가 수신될 때까지 sleep( )한다.

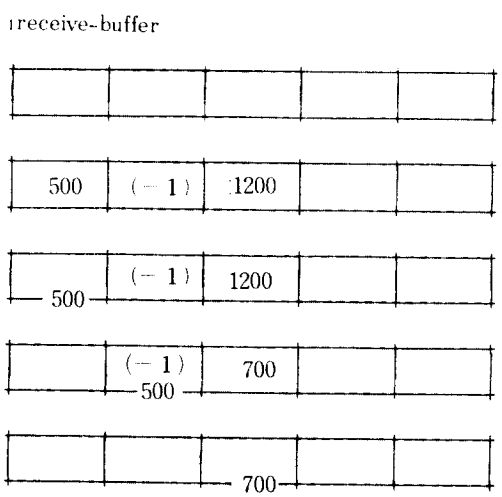


그림 4 read( )의 사용 예
Exawple (of read( )

1) ioctl( )로 size가 1000 byte인 receive-buffer를 4개 할당 받았다.

2) 상대방에게 write( )로 각 size가 500, 1200 byte인 packet을 두 개 보내왔다.

3) read(fd, buf, 1000)을 하면 첫 packet (500 byte)를 읽는다.

4) read(fd, buf, 500)을 하면 둘째 packet의 일부(500 byte)를 읽는다.

5) read(fd, buf, 1000)을 하면 둘째 packet의 나머지 (700 byte)를 읽는다.

최대 size만큼만 읽으므로 한번의 read( )로 packet 전부를 읽지 못하는 경우에는 다음번 read( )로 나머지를 읽을 수 있다.

read( )를 할 경우의 예를 그림 4 에 보였다.

• write( )

data 전송에 쓰인다. 형식은 다음과 같다.

write(fd, buffer, size) ;

전송할 data를 user buffer로부터 임시로 할당받은 buffer로 copy한다. 그리고 ntq( )를 통해 NIB에게 transmit-job command를 보내고 sleep( )한다. NIB는 command대로 data를 전송한다.

Ⅲ. UTP(User Transparency Provider)

가 . Job에 따른 UTP의 전송 방식

UTP는 INS system에서 user transparency를 제공하기 위해 3BNET을 통하여 node간에 file을 interactive하게 전송하거나 RCE을 가능하도록 하는데 job의 종류에 따라 전송 방식을 다음과 같이 3가지로 나눈다.

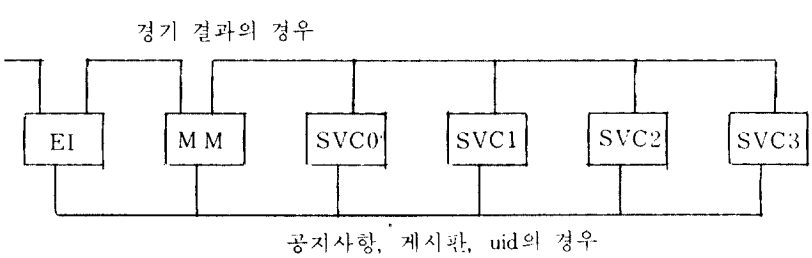


그림 5 broadcasting 방식
Broadcasting mode.

• broadcast 방식

INS system에서 모든 node가 공통으로 가지고 있는 data를 public data라고 하며, 경기결과, 공지사항, 게시판, uid 등이 여기에 속한다. broadcast란 이들 data를 각 node에 분배하는 경우인데, 어떤 node가 data를 처리하지 못하는 상황이 발생하면 consistency를 위해 MM node에 해당 job을 저장하였다가 추후에 복구한다.

• multicast 방식

INS system에서 어느 한 node에만 두는 data를 private data라고 하고 mail, form 등이 여기에 속한다. 그런데 어느 한 node가 down이 되면 이들 data를 사용할 수 없게 되므로 실제로는 각 node마다 backup node를 두고 original node의 data를 사용할 수 없게 되면 backup node의 data를 사용할 수 있도록 한다.

multicast 방식에서는 data를 송신하는 경우 original node와 backup node 모두가 정상이라면 두 node로 전송하고 하나가 이상이 있으면 정상인 node에 해당 job을 저장하여 추후에 pointcast를 이용하여 복구하도록 한다. data를 수신하는 경우에는 정상인 node중 하나를 선택하여 수신하면 된다.

• pointcast 방식

INS system의 어떤 node가 장애에 의해 service를 수행하지 못하다가(node down) 복구된 다음에 정상적으로 service를 수행하기 위해서는 down된 동안의 data를 recovery 하여 주어야 하는데, 이러한 경우 특정 node와 data를 송수신하거나 특정 node에서 command(RCE)를 수행시키고자 하는 경우에는 pointcast 방식을 이용한다.

나. UTP 관련 Process의 종류

UTP의 process는 requester와 deamon의 두 process 종류로 나누어 지는데, 전술한 UTP의 전송 방식에 따라 deamon은 다시 broad-deamon, multi-deamon, point-deamon의 3개로 나누어 진다.

• UTPR(requester process)

UTPR은 다른 프로세스로부터 요구되는 job

node name	ins 0	ins 1	ins 2	ins 3	ins 4	ins 5
original node	EI	MM	SVC 0	SVC 1	SVC 2	SVC 3
backup node	MM	EI	SVC 1	SVC 2	SVC 3	SVC 0

그림 6 original-backup 구성
Original-backup pairs

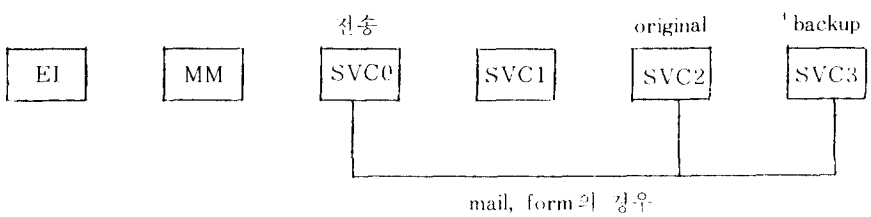


그림 7 multicasting 방식
Multicasting mode

에 따라 해당 deamon과 통신을 한다. UTPR의 address는 public data를 수신하는 process인데 모든 broad-deamon의 address는 같아야 한다.

• multi-deamon (multi & bmulti)

multi-deamon은 private data를 송신, 수신하는 process인데 서로 backup 관계에 있는 두 multi-deamon의 address는 같아야 한다. 모든 node마다 자신의 node를 위한 original multi-deamon (multi)과 backup node를 위한 backup multi-deamon (bmulti) 두개의 multi-deamon이 존재하게 된다.

• point-deamon (point)

point-deamon은 data recovery처럼 INS system의 service와는 무관하게 file을 송수신하고자 하는 경우와 RCE을 하는 경우가 되는데 point-deamon의 address는 전 node를 통하여 유일하도록 정하여 주어야 한다.

• server process (server)

deamon은 channel setup 과정에서 UTPR의 연결 요청에 대해 응답만을 하는 반면에, server Process는 deamon이 만들며, UTPR과 직접 data를 주고 받는 process이다. server의 address는 UTPR이 만든다.

이들 process의 address format은 그림 8과 같다. 3BNET (Ethernet) 에서는 같은 address를 가진 process는 모두 동일한 packet을 수신하게 된다. 따라서 한번의 전송으로 여러 node에서 여러 process가 동시에 수신할 수 있도록 addressing방식을 정하였다.

다. UTP의 control-type과 packet 구조

UTP에서는 channel의 setup/release, data/acknowledge의 전송 및 재전송을 하기 위해 5가지의 control-type을 가지고 있으며 각각의 용도는 다음과 같다.

• CONT (connect)

UTPR이 해당 deamon에게 연결을 요청하거나, deamon이 연결 요청에 대한 응답을 하거나, 연결을 해제하는 경우의 control-type 을 의미하는데 이의 status는 REQUEST, C-END, ACK, NACK의 4개가 있다.

• SFT (send-file-transfer)

* pid : process id
* nid : node number
* tim : time

process name	0	1	2	3	4	5	6	7
UTPR		1	'R'	(pid,	nid,	tim)	0	7
server		1	'BNP'	(pid,	nid,	tim)	0	0
broad-deamon	1	'B'	'R'	'O'	'A'	'D'	0	0
multi-deamon	1	'M'	'U'	'L'	'T'	nid	0	0
point-deamon	1	'P'	'O'	'I'	'N'	nid	0	0

그림 8 UTP process별 address format
Address formates of TTP processes

상대 node에 DATA를 전송할 경우의 control-type으로, 이의 status는 REQUEST, ACK, NACK의 3가지가 있게 된다.

• RFT(receive-file-transfer)

상대 node로부터 DATA를 수신할 경우의 control-type으로, 이의 status는 REQUEST, ACK, NACK의 3가지가 있게 된다.

• CMD(command)

상대 node에게 command를 수행시키고자 하는 경우의 control-type으로 이의 status는 REQUEST, ACK, NACK의 3가지가 있게 된다.

• DATA

SFT나 RFT의 packet이 두 node간에 성공적으로 교환된 후 실제 DATA를 전달할 경우의 control-type으로, 이의 status는 DATA, ACK, NACK의 3가지가 있게 된다.

UTP의 packet은 다음의 구조를 가지고 있다.

• connect-channel setup/release 요구시 사용된다.

```
struct connect {
        char control-type ;
        /* CONT */
        char status ;
        /* REQUEST, C-END */
        short node-number ;
        char utpr-address[8] ;
        char server-address[8] ;
        char job-id[8] ;
```

• data-file, command를 전송할 때 사용한다.

```
struct data {
        char control-type ;
        /* SFT, RFT, CMD, DATA
                                */
        char status ;
        /* REQUEST */
        short node-number ;
        char my-address[8] ;
        int size
        char data[2032] ;
```

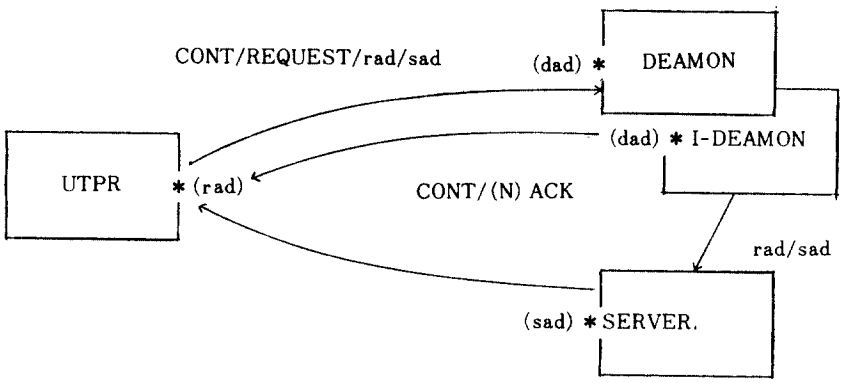
• ack-acknowledge packet으로 status가 NACK일 경우에는 error 항목에 그 이유가 들어간다.

```
struck ack {
        char control-type ;
        /* CONT, SFT, RFT, CMD, DATA
        */
        char status ;
        /* ACK, NACK */
        short node-number ;
        char my-address[8] ;
        int error ;
        char filler[12] ;
```

라. Channel Setup 방식

UTP를 통해 어떤 처리를 하려면, 반드시 channel을 setup한 다음 그 channel을 통하여 node간에 communication을 하게 된다. 여기서 channel setup은 UTPR과 server가 서로의 address를 알게되는 것을 의미한다. UTP의 channel setup 방식은 다음과 같다.

UTPR은 다른 process로부터 어떠한 요청이 있게 되면 요청을 분류하여, 우선 해당 요청에 맞는 deamon과 channel setup을 시도한다. 각 deamon의 address는 well-known address



* rad : UTPR-address
* sad : server address (generated by UTPR)
* dad : deamon-address (well-known address)

그림 9 Shannel setup 과정
Sequence of channel setup.

이므로 이 address를 통해 해당 deamon에게 자신의 address와 server의 address를 알리고 난 다음 연결 요청에 대한 응답을 기다린다. 이 응답이 ACK이면 server의 address를 통해 server와 직접 communication이 이루어지게 된다.

한편 deamon은 자신에게 연결 요청이 오면, 자신과 똑같은 intermediate deamon을 하나 만들어 연결 요청에 대한 나머지 작업을 시키고 자신은 다른 연결 요청을 기다린다.

intermediate deamon은 server를 만들어 UTPR의 address와 server의 address를 전달하고 UTPR에게 ACK를 보낸다. 위의 과정에서 error가 발생하면 NACK를 보낸다. server 는 지금부터 UTPR과 직접 communication 할 수 있다.

라. DATA의 송수신 및 RCE

UTPR과 server간에 channel이 setup되면 상호간에 file 전송을 통해 DATA를 주고 받을 수 있다. 그리고 server에게 command를 보내어 수행시킬 수도 있다. 이 때 사용되는 control -type은 SFT, RFT, DATA, CMD 등이다. 각각의 경우에 대해 UTPR과 server는 그림 10과 같이 packet을 주고 받는다.

마. UTPR과 다른 Application Process와의 Interface

Program에서 다음과 같은 구분으로 UTPR을 통해 원하는 작업을 할 수 있다.

```
if (fork( ) = = 0 )
  execl("utpr", " ", job-flag, s-file [, d-
        node, d-file], 0);
wait (& status) ;
```

여기서 s-file은 현 node에서의 file명, d-file은 상대 node에서의 file명이며, d-node는 상대 node명이다. fork( )는 child process를 만드는 것이며, execl( )은 그 process에 다른 process의 imeage를 overlay하고 실행시키는 것이다.

예를 들면, uid 등록 항목이 변경되는 경우, 모든 node의 uiddb가 똑같이 변경되어 각 uiddb의 consistency가 유지되어야 한다. 다음의 경우

```
execl("utpr", " ", "-u", uid-record, 0);
```

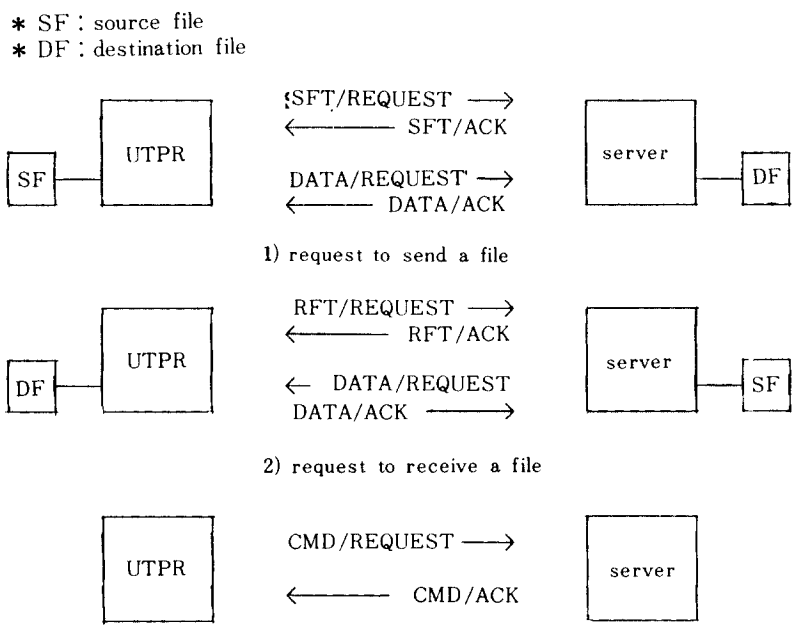


그림10 data의 송수신 및 RCE
Exchange of data and RCE.

job-name	job-flag			+	-	=	
bulletin	+-b	s-file	d-file	write	delete	·····	B
notice	+-n	s-file	d-file	write	delete	·····	B
uid	-u	s-file		·····	update	·····	B
game result	-g	s-file		·····	update	·····	B
form	+=f	s-file	d-node-d-file	write	·····	read	M
mail	+-=c	s-file	d-node-d-file	write	delete	read	M
file	-r	s-file	d-node-d-file	·····	receive	·····	P
file	-s	s-file	d-node-d-node	·····	send	·····	P
RCE	-x	s-file	d-node-d-file	·····	RCE	·····	P

그림11 UTPR의 argument
Arguments of UTPR.

UTPR은 SFT protocol을 이용하여 uid-record를 전송하고 CMD protocol을 이용하여 각 node에서 uid-update라는 process를 수행시킨다. uid-update는 수신한 uid-record에 따라 uiddb를 변경한다.

바. 운영 통계

response time을 user process와 UTP 사이에 intearfce가 시작되면서 끝날때까지 걸리는 시간이라고 하자. 86 아시안 게임 동안 UTP 의

job	전체 요구 회수	전체 전송 size	평균 response
RCE	1,697	0 Kbyte	5 sec
Send-File	1,723	5,897 Kbyte	4 sec
Rece ve-File	443	28,646 Kbyte	6 sec
Total	44,430	314,416 Kbyte	9 sec

그림12 UTP의 응답시간
Response time of UTP

response time을 그림 12에 보였다. user가 command를 입력한 때부터 UTP가 job을 수행하고 결과가 나타나기까지 약 4.7초가 소요되었다. 게임 기간 동안 각 node, application, 시간대 별로 통계를 내었으며 UTP의 response time은 time sharing environment하에서 system load에 따라 심한 편차를 보임을 알 수 있었다.

〔주〕 통계중 전체 전송 size는 Kbyte 단위로 truncate 하였으며 따라서 RCE의 경우 각 packet size가 1K byte 미만이므로 누계되어 0 Kbyte란 수치가 나왔다.

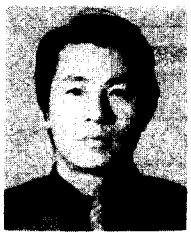
아시안 게임 기간 UTP는 request마다 평균 7.1 Kbyte를 전송하였으며 이에 걸린 시간은 9초 였다.

Ⅳ. 결 론

UTP는 86 아시안 게임 동안 6대의 3B 20S를 node로 하여 정상 동작을 하였다. 그러나 88 올림픽 때는 node수가 14로 늘어나고, load도 클 것으로 예상된다. 기존 UTP는 UTP에 대한 요구가 있을때마다 요구별로 매번 intermediate deamon과 server라는 새로운 process를 만들어야 하는데 이것이 단점이다. response time의 많은 부분이 channel setup 과정에서 process 생성에 소요되었고 response time이 system load에 따라 많이 좌우 되었으므로 좀더 빠른 channel setup 방식이 필요로 한다. LAN을통해 고속의 data 전송이 가능하므로 새로운 process의 생성 없이 IPC(interprocess communication)를 이용하여 job을 전달하는 방식도 검토되어야겠다.

參 考 文 獻

(1) Kerninghan, B., Pike, R., "The UNIX Programming Environment"
(2) Bach, M. J., "Design of the UNIX Operating System"
(3) Weitzman, C., "Distributed Micro/Minicomputer Systems"
(4) Tanebaum, A., "Computer Networks"
(5) Stallings, W. "Data and Computer Communications"
(6) GSS, "3Bnetwork Users/Operators Guide"
(7) DEC "Introduction to Local Area Networks"
(8) Metcalfe, R.M., Goggs, D.R., "Ethernet: Distributed Packets Switching for Local Computer Networks" Commun. ACM, vol. 19, July 1976.
(10) Lewan, D., Long, H., "The OSI File Server" Proceedings of the IEEE, December 1983.
(11) 한국데이타통신 "올림픽(아시안) 종합정보망 시스템 분석 명세서" '85. 5.
(12) 한국데이타통신 "올림픽(아시안) 종합정보망 시스템 설계 명세서" '85. 12.
(13) 한국데이타통신 "올림픽(아시안) 종합정보망 시스템 운영 지침서" 1986. 8.
(14) 한국데이타통신 "86 INS 운영보고서" 1986. 12.



李 哲 洙(Chul Soo LEE) 正會員
1945年 3 月20日生
1970～1972：서울大學校 文理科大學 數學科(學士)
1975～1977：韓國科學院 電算學科(碩士)
1977～1980：韓國科學院 電算學科(博士)
1972～1975：陸軍士官學校 教授部 數學科(專任講師)
1981～1982：서울地下鉄公社(研究委員)
1982～現在：韓國데이타通信(株)

黃 文 俊(Moon Joon HWANG) 正會員
1959年 7 月 7 日生
1978年 3 月～1982年 2 月：高麗大學校電子工學 學士
1982年 3 月～1985年 2 月：高麗大學院電子工學(컴퓨터 工學專攻)碩士
1985年 1 月～現在：韓國데이타通信(株)研究員

李 華 淵(Hwa Yeon LEE) 正會員
1952年10月20日生
1979年 2 月：東國大學校 電子計算學科
1984年：延世産業大學院電子計算學科
1979年～82年：中小企業銀行 行員
1982年～現在：韓國데이타通信(株)올림픽事業団 課長
1981年～現在：政府電子計算所公務員電算教育센터 講師