論文 86-23-1-1

Packet Switching에 의한 공중 Computer 통신망 개발 연구 -제3부: KORNET NNP의 X.25 Software 설계 및 구현

(Development of a Packet-Switched Public Computer Network - PART 3: X.25 Software Design and Implementation of the KORNET NNP)

崔埈均*, 金洛明**, 金炯淳*, 殷鍾官*, 任基弘*, 趙榮鍾*, 趙東浩*

(Jun Kyun Choi, Nak Myeong Kim, Hyung Soon Kim, Chong Kwan Un, Gi Hong Im, Young Jong Cho and Dong Ho Cho)

要 約

이 논문은 packet 교환 방식에 의한 공중 통신망 개발에 관한 4편의 논문중, 제3부 논문이다. 본 제3부 논문에서는 KORNET의 NNP에 packet mode data terminal equipment(DTE)를 접속하기 위한 X.25 protocol의 구현 과정에 대하여 기술한다.

먼저 system의 설계에 있어서 각 기능에 따라 software module을 구분하고 module간에 상호접속에 관한 규정을 마련한 뒤에 software structure를 결정하였다. 각 layer protocol은 state machine design 기법을 사용하여 설계되었으며, program의 coding을 위해서 specification 및 description language(SDL)과 program design language(PDL)을 기본으로 한 조직적인 개발기법을 사용하였다. 또한 각 module의 구동방법과 system performance를 분석하였다.

Abstract

This is the third part of the four-part paper describing the development of a packet-switched computer communication network named the KORNET. In this paper we describe the design and implementation of the X.25 protocol connecting packet mode data terminal equipments (PDTE's) with data circuit terminating equipments (DCE's).

In the KORNET, the X.25 protocol has been implemented on the line processing module-A (LPMA) of the network node processor (NNP). In the implementation of X.25, we have divided the software module according to the service function, and have determined the rules that interact between the modules. Each layer protocol has been developed using the technique of the finite state machine. Before the actual coding of softwares, we have used formal software development tools based on the specification and description language (SDL) and program design language (PDL) recommended by the CCITT.

In addition, for the efficient operation of the X.25 protocol system we have analyzed the system performance and the service scheduling method of each module. The results will also be given.

*正會員, 韓國科學技術院 通信工學研究室
(Communications Research Laboratory, KAIST)
**正會員, 金星電氣(株) 派遣
(Gold Star Elec. Co., Ltd)
接受日字: 1985年 6月 10日

I. 序 論

1960년대 말 ARPANET을 효시로 하여 computer 통신망이 구현된 이래 SNA, DECNET, TYMNET, TELENET 등 많은 통신망들이 개발되어 왔고 이러한 추세는 증가일로에 있다.[1]

Packet교환방식의 KORNET는 이러한 수요에 부응하여 개발되었는데, 개발된 통신망은 CCITT 권고사항을 전적으로 따라서 protocol들을 구현하였다. Computer 통신망의 개발 과정은 다른 통신시스템이나 computer system의 개발과 비슷한 단계를 거치게 된다. 먼저 통신망의 운영 방식과 통신망을 구성하는 각 network node processor(NNP)의 동작방법을 확정하고 접속하는 각종 단말기에 대하여 channel 서비스 방법을 결정한다.

서비스 사양이 결정되면 기능별 전체 구조를 확정하며 각 layer protocol의 구현방법과 인접되는 다른 layer사이의 접속규약을 정한다.

이러한 통신망을 구현함에 있어 고려되어야 할 사항으로는 먼저 접속될 다른 시스템과의 호환성(compatibility) 및 서로 다른 장소에서 여러 사람이 개발함으로 인한 protocol상의 mismatch 등이 있다. 또한 각 layer protocol은 표준규약을 완전하고 일관성있게 만족시키면서 개발해야 하며 multi-task 및 multi-processing 기술을 사용하여 real time processing에 알맞도록 단계적이고 조직적으로 개발해야 한다.

현재 일차로 구현된 KORNET통신망은 ARPANET의 interface message processor(IMP)와 유사한 3개의 NNP로 구성된다. 각 NNP는 네가지의 서로 다른 기능을 갖는 hardware board로 구성되며 각 board는 그 기능에 따라 독립적인 작업을 한다.

본 논문은 한국 최초로 개발된 packet 교환방식에 의한 computer 통신망 개발에 관한 제3부 논문으로서, 여기에서는 제1부의 KORNET의 개요와 NMC개발,[2] 제2부의 KORNET 설계 및 NNP개발에 관한 논문[3]에 이어 KORNET을 구성하는 NNP의 X.25 software 설계 및 구현에 관해 중점적으로 다룬다. NNP의 PAD 및 network management software 개발에 관한 자세한 사항은 이 논문에 이은 제4부 논문[4]에서 중점적으로 다루고 있다

본 제3부 논문에서는 제Ⅰ장의 서론에 이어 제Ⅱ장에서는 X.25 protocol 및 software 구조에 대하여 설명하고, 제Ⅲ장에서는 X.25 protocol을 구현하는 방법에 대하여 기술한다. 또한 제Ⅳ장에서는 X.25 protocol에 대한 performance를 분석하고 마지막으로 제Ⅴ장에서 결론을 맺는다.

Ⅱ. X.25 Protocol 및 Software구조

X.25 protocol은 공중 data network에서 packet mode로 동작하는 PDTE와 DCE사이의 접속을 규정한 protocol이며 국제표준연맹의 open systems interconnection(OSI)의 7계층구조로 볼 때 physical layer, link layer, packet layer의 3개의 layer architecture로 구성된다.

첫째로 physical layer는 단말장비와 NNP사이의 전송선로의 접속을 담당하며 이에 관한 CCITT 권고사항으로는 X.21과 X.21 bis가 있는데 KORNET에서는 X.21 bis에 상응한 EIA의 RS-232C 접속규약을 따른다. 둘째로 link layer는 frame format을 결정하고 transmission error의 detection 및 recovery를 담당하여 상위 layer에 대하여 error free channel을 제공한다. 셋째로 packet layer는 packet format를 결정하고 하나의 link를 여러 개의 logical channel로 multiplexing하며 packet sequence numbering을 사용하여 flow control 및 error control을 행한다. 또한 closed user group(CUG)의 가입, fast select의 선택 등과 같은 optional user facilities들을 제공한다.

X.25 protocol을 service하는 software의 구조를 보면 그림1과 같다. X.25 software에는 link layer 및 packet layer protocol을 수행하는 FRM-MAIN과 PKT-MAIN이 있고 physical layer와의 접속을 위해 high level data link control (HDLC) receiver 및 transmitter module이 있으며 이 밖에 system bus를 통하여 다른 hardware board와 접속을 위해 inter-station communication protocol (ISCP)가 있다.

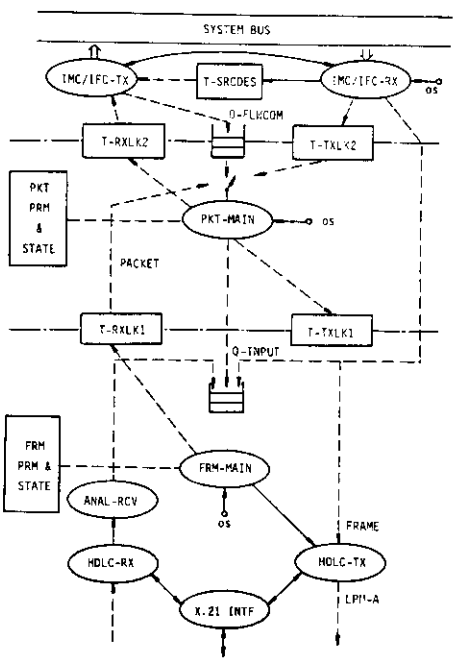


그림 1. X.25 software의 구조
Fig. 1. Software structure of X.25 protocol.

1. Link Layer

X.25 link layer protocol에 대한 frame format및 procedure는 CCITT 권고사항[5]에 자세히 기술 되어 있으므로 여기서는 KORNET에서 구현된 link layer protocol 체계에 대해서만 설명한다.

KORNET의 link layer 접속방법은 link access procedure balanced mode(LAPB)를 따르며 hardware 제어부와의 interface, packet layer software 의 interface 및 중앙 protocol 처리부로 나뉘어진다. 이중 중앙 protocol 처리부는 이른바 state machine design 기법이 도입되었으며, 이 module을 중심으로 software input/output(I/O), software timer등이 연동된다. 그림2는 link layer software의 구조를 나타낸 것이며 이는 크게 5개의 module로 나뉘어진다.

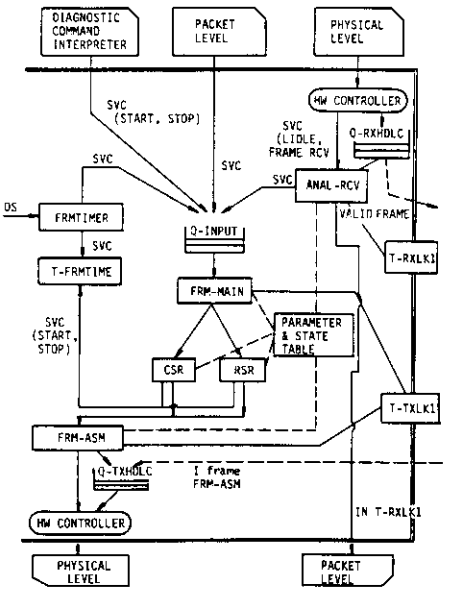


그림2. Link layer software의 구조
Fig. 2. Software structure of link layer.

첫째로, hardware receiver로 부터 수신된 개개의 frame을 일차적으로 분석하는 software input module인 ANAL-RCV가 있다. 여기서는 수신된 frame의 유효성을 검진한 후 이를 main buffer에 옮겨주고 수신된 frame에 대한 code를 input code queue에 담아주는 일을 행한다. 이 module은 operating system(OS)에서 제공하는 supervisory call(SVC)의 발행으로 기동된다.

둘째로, 중앙 protocol 처리부가 있는데, 이는 FRM-MAIN, CSR, RSR, FRM-ASM으로 구성된다. 먼저 FRM-MAIN은 input code queue로 부터 사건발생을 확인하고 parameter 및 state table로 부터 해당되는 값을 읽어 현재의 상태에 따라 적절한 조치를 수행하게 되는데 전송방향에 따라 command service routine (CSR)과 response service routine(RSR)로 나뉘어져 수행된다. CSR이나 RSR은 CCITT X.25 권고사항에 따라 적절히 protocol처리를 해주는데 처리 업무로는 관련된 system 변수의 update, 필요시 software timer의 기동명령 또는 정지명령의 하달, 상대국으로 전송해야 할 frame을 결정하여 FRM-ASM을 기동하는 일 등이 있다.

CSR과 RSR의 state는 16개의 protocol state로 구분되어 각각 16개및 14개의 입력사건에 대하여 처리한다.[6] 이는 크게 5개의 phase로 구분할 수 있는데 link가 차단된 상태인 DISCONNECTED phase, SABM 전송에 대하여 응답을 기다리는 DISC-ACK-WAIT phase, information frame을 전송하는 CONNECTED phase 및 link의 이상 상태 발생에 대하여 처리하는 EXCEPTIONAL phase이다. 여기서 CONNECTED phase는 frame 전송에 있어서 flow control, error detection 및 recovery를 위해 information transfer state, reject recovery state, time-out recovery state, busy state등의 4개의 작은 group으로 분리할 수 있다. 그림3은 CSR의 state transition diagram을 나타낸 것이며 RSR의 경우도 이와 유사하다. 끝으로 FRM-MAIN은 전송할 frame을 실제 조합하여 hardware transmitter로 전송명령을 하달한다.

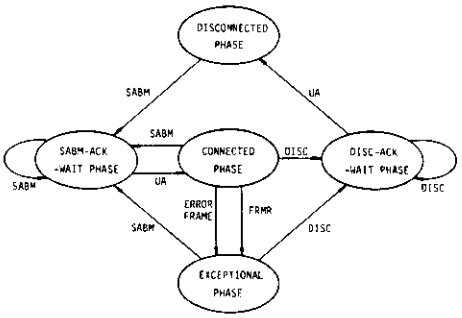


그림3. Link level의 command service routine (CSR)의 state 변환 도형
Fig. 3. State transition diagram of command service routine (CSR) at link level.

셋째로, protocol 규정의 하나인 timer module로 FRM-TIMER가 있는데 이는 software timer로구현되어 timer의 기동명령 및 정지명령을 수행한다. FRM-TIMER는 OS의 real time clock에 의해 주기적으로 기동되어 현

재 동작중인 timer를 update하고, timer-out 여부를 검사하여 time-out 발생시 여기에 상응한 code를 만들어 input code queue에 쌓아준다.

넷째로, hardware receiver module로 $HDLC_{rx}$ 가 있는데 여기서는 실시간 처리가 특히 강조되며 실제 hardware port(SIO port)로 부터 character 단위로 수신되는 frame을 받아서 지정된 queue에 쌓아주는 업무를 수행한다. 그림 4 에서 $HDLC_{rx}$ module은OS에 의하여 기동되어 hardware chip을 운용하며 data가 수신되면 Q-RXHDLC queue에 쌓아주고 ANAL-RCV에게 알린다.

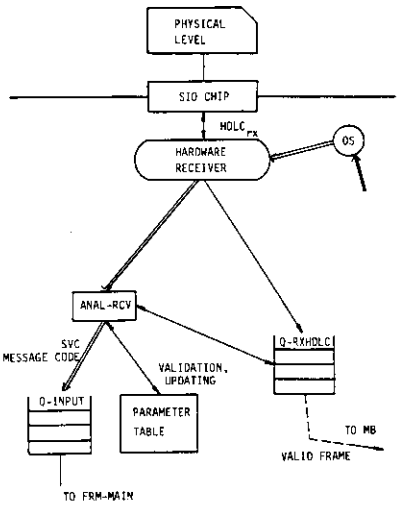


그림 4. Hardware receiver (HDLCrx)의 주변
Fig. 4. The environment of hardware receiver (HDLCrx).

다섯째로 hardware transmitter module로 $HDLC_{tx}$ 가 있는데, 이는 Q-TXHDLC queue에 담겨진 frame을 character 단위로 내보내는 업무를 수행한다. 그림 5 의 $HDLC_{tx}$ module은 FRM-ASM으로 부터 전송할 frame이 있을 때 hardware chip의 운용상태를 검사하여 tx-enable 상태가 되면 한 character씩 hardware port로 송신한다. 이 밖에 link layer의 frame format과 control field format은 CCITT 권고사항을 따른다.[5]

2. Packet Layer

Packet layer는 신뢰성있고 효율적인 data communication을 위해 data channel의 multiplexing, 각 data 흐름에 대한 flow control 및 error control, 그 밖에 channel상에 중대한 error가 발생했을 때 communi-

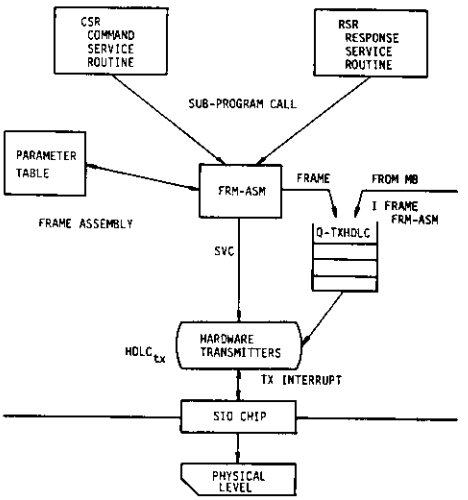


그림 5. Hardware transmitter (HDLCtx)의 주변
Fig. 5. The environment of hardware transmitter (HDLCtx).

cation path를 reinitialize 하는 reset 및 restart 기능등을 수행한다. 먼저 channel multiplexing을 하기 위해 logical channel identifier를 두며, flow control

표 1. Packet 그룹 및 기능
Table 1. Packet groupings/functions.

Packet Group	Function	Packet Types
Call setup and call clearing	Establish and terminate a virtual call for DTE/DCE communication; may convey data for higher-level entity processing	CALL REQUEST INCOMING CALL CALL ACCEPTED CALL CONNECTED CLEAR REQUEST CLEAR INDICATION CLEAR CONFIRMATION
Data and interrupt	Convey data or interrupt information for higher-level entity processing	DATA INTERRUPT INTERRUPT CONFIRMATION
Flow control and reset	Control the flow of data packets across a DTE/DCE interface	RECEIVE READY RECEIVE NOT READY REJECT RESET REQUEST RESET INDICATION RESET CONFIRMATION
Restart	(Re)Initialize communication between a DTE and a DCE	RESTART REQUEST RESTART INDICATION RESTART CONFIRMATION
Diagnostic	Pass error diagnostics to a DTE	DIAGNOSTIC

과 error control을 위해 send 및 receive sequence number와 flow control packet을 사용한다. 그리고 communication path의 reinitialization을 위해 control packet을 사용하는데, 표 1 은 packet layer의 기능별 group과 packet type을 나타낸 것이다.

KORNET의 packet layer는 접속되는 상대에 따라 2 가지의 기능으로 구별되는데 외부의 PDTE와 접속될 때는 DCE 기능을 갖게되며 network 내부의 다른 NNP와 연결될 때는 data switching exchange (DSE)의 기능을 갖는다. NNP와 NNP 사이에 DSE-to-DSE의 접속이 이루어질 경우에 내부 접속 규약에 의해 하나가 DTE mode로, 다른 하나가 DCE mode로 동작한다. Packet layer의 software 구조는 link layer와 유사하게 5 개의 module로 구성되며 그림 6 은 packet layer의 software 구조를 나타낸다.

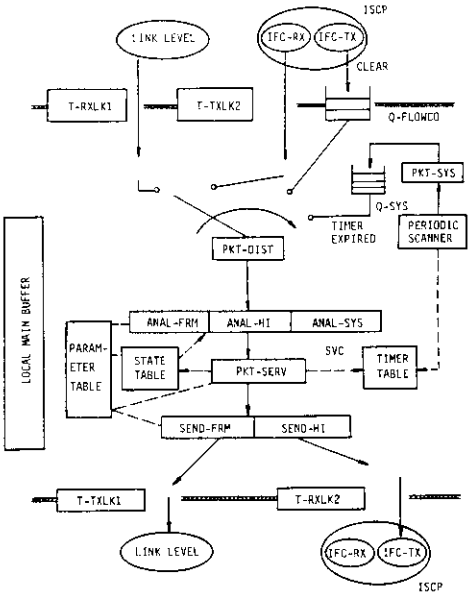


그림 6. Packet layer의 software 구조
Fig. 6. Software structure of packet layer.

첫째로 Packet layer로 입력되는 packet을 받아들이는 PKT-DIST는 NNP 외부에서 부터 수신되는 packet를 받아들이기 위해 link layer와 접속되고, NNP 내부의 다른 hardware board로 부터 수신되는 packet를 받아들이기 위해 interstation communication protocol (ISCP)와 접속된다. 그리고 protocol 상의 time-out 발생은 Q-SYS queue를 통해서 입력된다.

둘째로, 수신된 각 packet에 대하여 logical channel 별로 channel parameter와 state에 따라 분석 하는 ANAL-FRM, ANAL-HI, ANAL-SYS가 있다.

셋째로, 중앙 protocol 처리부인 PKT-SERV가 있는데 여기서는 logical channel별로 parameter table을 update하고, software timer의 기동 및 정지명령을 내리며, 다른 layer로 packet을 전송하기 위해 SEND-HI 및 SEND-FRM을 기동한다. 이러한 13개의 protocol state로 구분되어 있는 PKT-SERV는 여러 입력사건을 처리하며[6] 3 단계의 계층적 구조를 갖고 있다. 먼저 packet layer에는 initialize 하는 restart state들로 r1, r2, r3 state가 있고, 다음으로 logical channel 별로 virtual call service를 하는 call setup 및 clearing state 들로 p1 에서 p7 state까지 7개의 state가 있다.

또한 각 logical channel의 reinitialization을 위한 reset state들로 d1, d2, d3 state가 있다. 표2는 packet layer의 state name과 symbol을 나타낸다. 이러한 state 들은 state machine으로 운영되며 state diagram은 CCITT X.25 권고사항과 유사하다.[6]

표 2. Packet layer의 state 이름과 기호.
Table 2. State names and notation of packet layer.

DSE (DTE mode)		DCE or DSE (DCE mode)	
State name	Symbol	State name	Symbol
Restart States			
PACKET LEVEL READY	r1	LEVEL 3 READY	r1
DTE RESTART REQUEST	r2	AWAITING RESTART CONFIRMATION	r3
DCE RESTART INDICATION	r3	PROCESSING RESTART INDICATION	r2
Call setup and call clearing states			
READY	p1	READY	p1
DTE CALL REQUEST	p2	AWAITING CALL ACKNOWLEDGEMENT	p3
DCE INCOMING CALL	p3	PROCESSING INCOMING CALL	p2
DATA TRANSFER	p4	DATA TRANSFER	p4
CALL COLLISION	p5	CALL COLLISION	p5
DTE CLEAR REQUEST	p6	AWAITING CLEAR CONFIRMATION	p7
DCE CLEAR INDICATION	p7	PROCESSING CLEAR INDICATION	p6
Reset states			
FLOW CONTROL READY	d1	FLOW CONTROL READY	d1
DTE RESET REQUEST	d2	AWAITING RESET CONFIRMATION	d3
DCE RESET INDICATION	d3	PROCESSING RESET INDICATION	d2

넷째로 node 내부와 외부로의 packet을 송신하기 위해 SEND-HI, SEND-FRM이 있고 마지막으로 protocol timer로 PKT-SYS가 있다.

Packet layer와 인접되는 다른 module로는 link layer 외에 NNP 내부의 다른 hardware board와의 접속을 위해 ISCP가 있는데 이는 기능에 따라 interfunction

communication protocol (IFCP)와 intermodule communication protocol (IMCP)의 2층 구조로 되어 있으며 자세한 것은 제 4 부 논문에서 기술한다.

KORNET에서 구현된 packet layer의 packet format은 CCITT 권고사항의 modulo 8 의 경우를 따른다. 여기서 call request 및 incoming call packet의 DTE address는 CCITT X. 121 규정을 따르며[5] KORNET 내부 address로는 8 digit를 사용한다.

그 밖에 logical channel number 할당 방법 및 optional user facilities 사용과 관련된 field format 에 관한 자세한 사항은 제 4 부 논문에서 다룬다.

Ⅲ. X. 25 Protocol의 구현

제Ⅱ장의 X. 25 protocol과 software structure 로부터 system의 개발에 이르기 까지는 여러 단계의 작업이 필요하다.

첫째로, 각 layer protocol에 대한 software 사양으로 부터 전체 system의 서비스 방법을 결정한다.

둘째로, protocol 상의 미확정된 사항에 대한 처리 방법을 결정하고 주고받을 queue나 table structure를 정한다.

셋째로, 실제 program coding 작업을 행함에 있어 설계된 protocol과 program 사이에 논리적인 일관성을 유지하고 또 이를 용이하게 검증할 수 있도록 한다.

마지막으로 system의 test 및 debugging에 있어서는 각 module의 동작이 protocol과 일치하는지의 여부를 조사하고 특히 real time service에 있어 각 layer 간의 service synchronization이 잘 이루어지는 가를 검사한다.

제Ⅲ장에서는 이러한 단계적인 개발 방법에 따라 X. 25 protocol의 검색방법, 각 service task의 구동방법, flow control 및 protocol parameter 선정 방법에 대하여 기술한다.

1. Protocol 검증방법

일반적으로 protocol 설계에서 발생하는 잘못은 state deadlock, 규정되지 않은 상태의 발생, 수행되지 않는 state의 존재, 불확실한 state의 정의 등 네가지 종류로 구분할 수 있다. 이러한 오류를 검출하고 protocol을 검증하기 위해 보통 두 가지의 접근방식을 취하는데 이들은 reachability analysis와 program proofs이다.[9]

Reachability 분석이란 각 계층 protocol에 대하여 state간의 가능한 모든 상호작용을 조사함으로써 위의 4 가지 잘못을 검출하는 방법이다. 먼저 초기 state를 정하고 여기서 가능한 모든 state transition을 발생시켜 현재 구현된 state외에 새로운 state가 발생하는 지의 여부와 상호작용이 없는 state들의 존재 여부를 조사한다. 이러한 분석방법은 protocol 수행상의 가능한 모든 결과를 예측할 수 있으며 이를 통해 deadlock state나 terminating state들의 발생을 검출해 낼 수 있다. 이 방법은 state간의 상호작용이 복잡할 경우 "state space explosion" 이 일어날 수 있으나 state들 간에 hierarchy를 두거나 여러개로 grouping을 함으로써 피할 수 있다.

Program proof를 통한 protocol의 test는 coding 된 program이 서비스 규정을 만족하는가를 조사하는 것으로 protocol이 state machine design 기법으로 표현되어 있을 경우 CCITT에서 권장하는 specification 및 description language (SDL)과 program design language (PDL)의 단계적인 개발 기법을 사용하면 각 service의 흐름을 쉽게 알 수 있어 debugging이 용이하다.[7,8]

2. Task Process의 선정 및 구동방법

각 software module을 구동시킬 task process의 선정은 각 channel에 대하여 공정하고 효율적인 service가 이루어지도록 module의 service speed requirement와 service 방법에 따라 알맞게 선정해야 한다. 또한 service에 있어 우선권이 주어져야 하는 module과 그렇지 않은 module은 다른 task로 분리함으로써 구동상의 효율을 기해야 한다. 이때 불필요하게 task로 분리할 경우 발생되는 interface overhead를 고려해야 한다.

NNP에 구현되는 X. 25 service system은 operating system (OS) 위에 ANAL-RCV, FRM-MAIN, PKT-MAIN, IFC-RX, FRM-TIMER 및 PKT-SYS를 구성하는 software timer 등 모두 5 개의 user task process로 구성되며 그 밖의 다른 module들은 이들 task process의 종속 module로 동작한다.

각 task의 service 방식을 보면 먼저 PKT-MAIN은 basic task process이며 polling 방식으로 입력된 사건을 처리한다. 또한 software timer task는 OS에 의해 주기적으로 구동되며 이때 구동주기를 timer의 기본단위로 사용한다. 다음으로 ANAL-RCV, IFC-RX, FRM-MAIN task process들은 event driven 방식으로 구동된다. 각 software module 간의 interface 는 queue 나 table을 사용하거나, subroutine call 또는 OS의 system call을 이용한다.

한편 각 task process의 service priority 및 service scheduling 방식을 보면 다음과 같다. Real time

service를 요하고 random하게 구동되는 ANAL-RCV와 IFC-RX는 highest priority를 갖고 preemptive service를 행한다. 다음으로 service time이 작고 구동주기가 내부동작에 사용되는 software timer task는 second priority를 준다.

마지막으로 PKT-MAIN과 FRM-MAIN task는 복잡한 protocol service를 하는 데 각 communication path 당 한개의 information을 처리하기 위해서 이들 task는 같은 service rate를 유지해야 함을 그림 7의 결과로부터 알 수 있다.

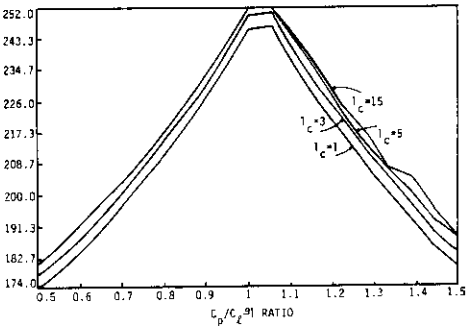


그림 7. Link layer와 packet layer의 처리용량의 비율에 따른 link throughput (l_c는 logical channel 수임)

Fig. 7. Link throughput versus the ratio of link layer to packet layer processing capacity.

3. Flow Control 및 Parameter 결정

X.25 protocol에서의 flow control은 sliding window 방식을 사용하며 piggybacked acknowledgement를 병용한다. Window size는 CCITT 권고 사항을 따라 link layer는 7, packet layer는 2로 한다. Channel의 흐름을 조절하기 위해 각 layer protocol은 자국의 busy 상태를 link layer와 packet layer에서 사용하는 buffer 수에 의하여 발생 시키는데 이때 그림8과 같이 hysterisis를 둔다. 여기서 B_L, B_H 값은 link layer 및 packet layer에서 이용가능한 최대 buffer 수에 대한 threshold를 고려하면서 각 logical channel에서 처리지연 시간이 작고 link가 최대 성능을 얻도록 결정하여야 한다. 그림 9를 보면 link capacity가 64Kbps, 각 logical channel의 전송속도가 9.6Kbps일 때 default window size에서 active logical channel 수가 10개 정도일때 부터 link가 최대성능을 발휘함을 알 수 있다. 이러한 결과를 통해 KORNET에서는 packet layer window가 2일때 각 link 당 active logical channel수를 10개라 보고 B_L과 B_H를 각각 18과 22로 두었다. 여기서 B_L과 B_H 값의 차이가 너무 작으면 발진이 일어나 throughput이 감소될 수 있다. System 내의 buffer 수가 충분할 경우 더 B_L과 B_H 값을 크게 둘 수 있으나 전송속도는 별로 향상되지 않는다.

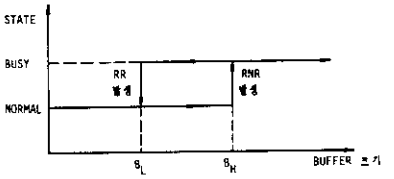


그림 8. Buffer 크기에 따른 흐름 조정 방법

Fig. 8. Flow control scheme based on the variation of buffer size.

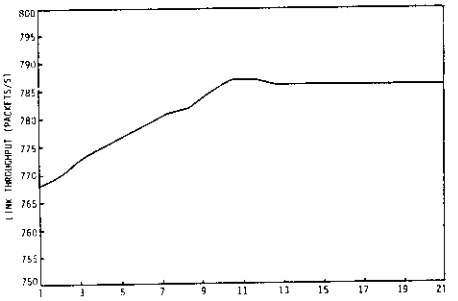


그림 9. 최적 service scheduling에서 logical channel 수에 따른 link throughput

Fig. 9. Logical channel number versus link throughput for the optimal service schedule.

이렇게 buffer의 크기에 hysterisis를 두어 channel busy 상태를 발생시키는 것외에 system 내에 전체 사용된 buffer 수에 대하여 2개의 threshold level을 두어서 1단계 threshold level을 넘으면 channel에 busy 상태를 발생시키고 2단계 threshold level을 넘으면 system의 deadlock 현상을 막기 위해 입력된 packet을 버린다.

Flow control과 더불어 channel error가 발생시 이를 recovery하기 위해 timer를 두는데 time-out 값은 다른 network에서 사용하는 값과 비슷한 값을 선정했다.[10,11,12] Link layer의 T1, T2 timer의 time-out 값을 각각 3초, 0초로 두었으며 재전송 허용횟수(N2)는 10으로 하였다. 그리고 packet layer의 time-out값은 CCITT X.25 권고규정을 따랐다.[5]

Ⅳ. Link Layer와 Packet Layer의 성능분석

여기서는 X.25 protocol를 구현함에 있어 여러 design parameter들이 system performance에 어떤 영

향을 미치는 지를 분석하기 위해 각 design변수, 각 layer의 throughput 및 transmission delay 사이의 관계식을 유도하고자 한다.

먼저 link layer의 throughput을 T_e, mean frame transmission delay를 t_v라 두면

$$T_e=\frac{E(\tilde{I})}{t_v} \tag{6}$$

가 된다. 여기서 $E(\tilde{I})$을 mean frame length(bits)이다. t_v를 구하기 위해 information frame의 mean transmission delay를 t_I(s), link의 modulus를 M, 전송된 frame의 응답시간을 t_{ack}(s) 라 하자.
또 frame error probabity를 P_B라 두면

$$t_v=t_I+P_BE[T_1(\omega)]+\frac{P_B^2}{1-P_B}\cdot t_2 \tag{7}$$

가 된다.[13] 여기서 $E[T_1(\omega)]$는 window가 ω일때 전송된 information frame이 파괴됐을 경우 재전송을 시작할 때 까지의 평균시간이다. t_2는 재전송되는 packet들 사이의 시간이다. 먼저 $t_{ack}\leq(M-2)\,t_I$인 경우에 $T_1(\omega)$를 구한다. Frame의 time-out 시간을 t_{out}(s)라 할때

$$C=Min\left\{M-2,\left\lfloor\frac{t_{out}}{t_I}\right\rfloor+1\right\}$$

이라 한다. 이때 $[r]$는 r를 초과하지 않는 가장 큰 정수이다. 여기서 $0\leq\omega<C$ 일때 $T_1(\omega)$는

$$T_1(\omega)=\begin{cases}(\omega+1)\,t_I+t_{ack}+t_I\\ (\text{단, } (\omega+1)\,t_I+t_{ack}<C\,t_I)\\ (\omega+1)\,t_I+\left\{\left\lfloor\frac{t_{ack}}{t_I}\right\rfloor+1\right\}t_I+t_I\\ (\text{단, } t_{out}\leq C\,t_I)\end{cases} \tag{8}$$

이고, $\omega=C$ 일때는

$$T_1(C)=\begin{cases}C\,t_I+t_S+t_{ack}+t_I\\ (\text{단, } t_{out}\leq C\,t_I)\\ t_{out}+t_S+t_{ack}+t_I\\ (\text{단, } t_{out}>C\,t_I)\end{cases} \tag{9}$$

가 된다.[13] 여기서 t_S는 frame이 supervisory frame일 때의 전송시간이다. 또한 $E[T_1(\omega)]$를 구하면

$$E[T_1(\omega)]=\sum_{w=\phi}^{C-1}(1-P_B)P_B^wT_1(\omega)+P_BT_1(C) \tag{10}$$

가 된다. 다음으로 t_2를 구하면

$$t_2=\begin{cases}C\,t_I+t_S+t_{ack}+t_I\\ (\text{단, } t_{out}\leq C\,t_I)\\ tout+t_S+t_{ack}+t_I\\ (\text{단, } t_{out}>C\,t_I)\end{cases} \tag{11}$$

가 된다.

여기서 식 (10), (11)을 식(7)에 대입하면 mean frame transmission delay t_v를 구할 수 있다. 이를 식 (6)에 대입하면 link throughput이 계산된다.

다음으로 packet layer throughput을 T_P, mean packet transmission delay를 t_v'라 하면

$$T_P=\frac{E(\tilde{I}')}{t_v'} \tag{12}$$

가 된다. 여기서 $E(\tilde{I}')$는 평균 packet length(bits)이다. t_v'를 구하기 위해 packet window size를 W, packet transmission time을 t_I', packet response time을 t_{ack} 라 두면 t_v'는

$$t_v'=t_I'P[\omega<0]+t_{ack}\;P[\omega=0] \tag{13}$$

가 된다. 윗 식에서 $P[\omega<0]$는 상대 node에서 packet을 받았을때 receive window가 0보다 클 확률이고, $P[\omega=0]$은 receive window가 0일 확률이다. 식(13)에서 t_I', t_{ack}는 각각

$$t_I'=E(\tilde{I}')/C_P \tag{14}$$

$$t_{ack}'=E(\tilde{I}')/C_P+2P+\frac{P_2E(\tilde{I}'^2)}{2E(\tilde{I}')C_P}+(1-\rho_2)\frac{l_a}{C_P}+\rho_2\frac{E(\tilde{I}')}{C_P} \tag{15}$$

로 표시된다.[14] 이때 C_P[bits/s]는 packet layer의 effective processing capacity이며, P는 node processing delay이고, l_a[bits]는 logical channel에 전송되는 flow control packet의 length이다. 또한 ρ_2는 logical channel의 수신방향의 utilization이며 식(15)에서 $E(\tilde{I}^{12})/2E(\tilde{I}')C_P$는 channel의 remain serviec time이다. 다음에는 $P[\omega<0]$, $P[\omega=0]$를 구해본다. Packet layer가 sliding window flow control 이면서 piggybacked acknowledgement를 한다고 가정하면 상대 node의 receive window process는 그림10과 같이 나타난다. 그림 10에서 receive window가 i일 확률을 P_i라 두면 평형상태에서 P_0는

$$P_0=\begin{cases}\dfrac{\rho_2}{\rho_2+\left\lfloor\dfrac{(1-\rho_2)\rho_1+\rho_2}{(1-\rho_2)\rho_1}\right\rfloor\omega-1} & (\text{단, } 0<\rho_2\leq1)\\ \dfrac{\rho_1}{\rho_1+\omega-1} & (\text{단, } \rho_2=0)\end{cases} \tag{16}$$

가 된다. 또한 $P[\omega=0]$는

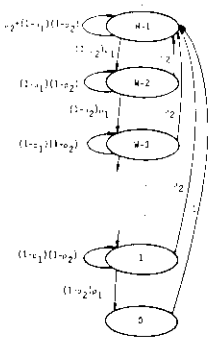
$P[\omega=0]=P_0|\rho_1=$,

이므로 $P[\omega=0]$와 $P[\omega>0]$는

$$P[\omega=0]=\begin{cases}\dfrac{\rho_2(1-\rho_2)^{w-1}}{1-(1-\rho_2)^w} & (\text{단, } 0<\rho_2\leq1)\\ \dfrac{1}{\omega} & (\text{단, } \rho_2=0)\end{cases} \tag{17}$$

$$P[\omega>0]=1-P[\omega=0]=\begin{cases}\dfrac{1-(1-\rho_2)^{w-1}}{1-(1-\rho_2)^w} & (\text{단, } 0<\rho_2\leq1)\\ \dfrac{\omega-1}{\omega} & (\text{단, } \rho_2=0)\end{cases} \tag{18}$$

이 된다. 식(14), (15), (17), (18), 을 식(13)에 대입하면 mean packet transmission delay t_v' 를 구할 수 있다. 이 t_v' 를 식(12)에 대입하면 packet throughput을 얻을수 있다.



수 μ_1 : Packet layer의 receive utilization
μ_2 : Packet layer의 send utilization

그림10. Packet layer와 receive window process
Fig. 10. Receive window process of packet layer.

V. 結 論

본 제3부 논문에서는 공중 data network에서 가장 기본적인 접속 protocol인 X.25의 구현방법에 대하여 기술하였다.

먼저 protocol 규약으로 부터 각 service function 들을 분리하는 과정을 밝혔다. 또 분리된 각 module 들 간의 접속방법 및 각 layer protocol 간의 data 전송방법, hardware chip을 통한 data의 송, 수신 방법들을 정함으로써 전체 software 구조를 결정하였다. 이때 node의 운영방법은 system의 modularity, 확장성 문제 뿐만 아니라 개발의 용이성 및 test의 편리함 등을 고려하였다.

또한 real time processing system으로서의 system performance 문제를 고려하여 multi-task 및 multi-processing 기법을 사용했다.

각 layer protocol의 구현에 있어서는 state machine design 기법을 이용하여 protocol을 설계하고 SDL과 PDL을 사용한 program coding을 통하여 복잡한 service를 하는 protocol software를 조직적으로 개발하였다.

이 KORNET의 각 NNP는 통신망의 구조에 관계없이 독립적으로 운영할 수 있는 distributed communication network이다. 또한 NNP는 각 기능별로 독립적으로 운영할 수 있는 구조를 갖고 있기 때문에 개발 및 test 그리고 확장이 용이할 뿐만 아니라 유사한 communication system의 구현에 쉽게 응용할 수 있다.

參 考 文 獻

[1] A.S. Tanenbaum, *Computer Networks*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.

[2] 은종관 외,"Packet Switching에 의한 공중 Computer 통신망 개발연구－제1부 : KORNET의 개요와 Network Management Center (NMC)의 개발," 대한전자공학회지, 第22卷 6號, 1985年 11月.

[3] 은종관 외,"Packet Switching에 의한 공중 Computer 통신망 개발연구－제2부 : KORNET의 설계 및 Network Node Processor (NNP) 개발," 대한전자공학회지, 第22卷 6號, 1985年 11月.

[4] 은종관 외,"Packet Switching에 의한 공중 Computer 통신망 개발연구－제4부 : KORNET NNP의 Packet Assembler/Disassembler (PAD) 및 Network Management Software의 구현," 대한전자공학회지, 第23卷 1號, 1986年 1月.

[5] CCITT Recommendation X. 25, X. 121, 1980.

[6] Packet Switching에 의한 Computer 통신망 개발 1차년도 최종보고서, 한국과학기술원, 1983.

[7] CCITT Recommendation on the functional specification and description language (SDL) - Recommendation Z. 100 to Z. 104, 1980.

[8] S.H. Caine, and E.K. Gordon,"*PDL-A Tool for Software Design*", Proc. of National Computer Conf., 1975.

[9] G.V. Bochmann, and C.A. Sunchine, "Formal methods in communication protocol design," *IEEE Trans. Commun.*, vol. COM-28, no. 4, April 1980.

[10] American Telephone and Telegraph Company, BX. 25, 1982.

[11] X. 25/X. 75 Interface Reference Manual Tymshare Inc., 1981.

[12] Telenet X. 25 Documentation Service, GTE Telenet Communications Corp., 1979.

[13] W. Bux, K. Kümmerle, and H.L. Truong, "Balanced HDLC procedures: A performance analysis, *"IEEE Trans. Commun.*, vol. COM-27, no. 5, May 1979.

[14] L.W. Yu, and J.C. Majithia, "An analysis of one direction of window mechanism," *IEEE Trans. Commun.*, vol. COM-27, no. 5, May 1979. *