論文 85-22-6-18

Packet Switching에 의한 공중 Computer 통신망 개발 연구 -제 2 부 : KORNET의 설계 및 Network Node Processor (NNP)의 개발

(Development of a Packet-Switched Public Computer Communication Network -PART2 : KORNET Design and Development of Network Node Processor (NNP))

趙有濟*, 金熙東*, 殷鍾官*, 李貞律*, 李孝珍*, 鄭忠教*, 趙威德*, 趙熒濟*

(You Ze Cho, Hee Dong Kim, Chong Kwan Un, Jeong Rhul Lee, Hyo Jin Lee, Choong Kyo Jeong, Wi Duck Cho and Hyung Je Cho)

要 約

이 논문은 packet switching 방식에 의한 공중 computer 통신망 개발에 관한 4편의 논문중 제 2 부의 논문으로 제 1 부의 KORNET의 개요 및 network management center (NMC) 개발에 관한 논문에 이어 KORNET의 설계와 network node processor (NNP)의 개발에 대해 기술한다. KORNET은 3개의 NNP와 하나의 NMC 로 일차 구성하였는데, NNP는 MC68000 microprocessor를 이용한 multiprocessor system으로 구현되었고, NMC는 중형 computer인 MV/8000 system을 사용하여 개발하였다. KORNET에서의 packet service 방식은 virtual circuit(VC) 방식으로 하고 routing은 node나 선로의 상태변화에 쉽게 대처할 수 있는 분산적응방식(distributed adaptive routing)을 사용하였다. 또한, buffer management는 dynamic sharing 방식을 채택하여 storage의 사용에 대한 효율성을 높였다. NNP system의 hardware는 modularity를 고려하여 확장이 용이하게 하였으며, software는 CCITT 권고사항 X.25, X.3, X.28, X.29 등을 따라 구현하였다.

Abstract

This is the second part of the four-part paper describing the development of a packet-switched computer network named the KORNET. In this paper, following the first part paper that describes the concepts of the KORNET and the development of the network management center (NMC), we present the design of the KORNET and the development of the network node processor (NNP). The initial configuration of the KORNET consists of three NNP's and one NMC. We have developed each NNP as a microprocessor-based (MC68000) multiprocessor system, and implemented the NMC using a super-mini computer (MV/8000. For the KORNET we use the virtual circuit (VC) method as the packet service strategy and the distributed adaptive routing algorithm to adapt efficiently the variation of node and link status. Also, we use a dynamic buffer management algorithm for efficient storage management. The hardware of the NNP system has been designed with emphasis on modularity so that it may be expanded esily. Also, the software of the NNP system has been developed according to the CCITT recommendations X.25, X.3, X.28 and X.29.

*正會員, 韓國科學技術院 通信工學研究室
(Communications Research Laboratory, KAIST)

接受日字 : 1985年 6月 10日

I. 序 論

1960년대말 최초의 packet 교환 방식을 이용한 computer통신망인 ARPANET이 개발된 후, 70년대에 들

어 세계 각국에서 실험 통신망 형태로서 프랑스의 RCP, 영국의 EPSS, 일본의 DDX등이, 상업용 통신망으로는 캐나다의 DATAPAC, 미국의 TELENET등이 개발되었다.

이와같이 각국에서 computer통신망을 개발하는 주요한 이유는 분산되어 있는 computer 및 주변기기, data base 등과 같은 다양한 resource들로 통신망을 구성함으로써 여러명의 사용자들이 공유하여 이용하게 되어 효율성을 높이고, 하나의 resource가 고장일 경우 back-up기능이 가능하며, 사용자들 간의 정보교환이 가능하기 때문이다.[1]

Data traffic의 특성은 음성과는 달리 bursty한 성격이 강하고, 약간의 전송지연은 허용되나 error가 없어야 하며, 송·수신에 있어서 처리 속도의 matching을 위해서 flow control이 필요하다. 이러한 data traffic의 특성 때문에 packet 교환방식을 이용하면, 종래의 circuit 교환방식에 비해 statistical multiplexing이 가능하여, channel이나 buffer, switching processor등의 효율적인 dynamic sharing을 통해 비용 및 성능면에서 많은 잇점이 있다.[2],[3]

Packet 교환방식은 packet service방식에 따라 크게 virtual circuit(VC) 방식과 datagram(DG) 방식으로 구분되는데,[3],[4] 이들 두 방식은 장단점이 있으나, DG 방식에 비해 VC방식은 network내부에서 flow control이 용이하고, packet의 중복, 누락 및 손상현상을 subnetwork에서 방지할 수 있다. 또한, VC방식은 packet의 전송 순서가 일정하여, 일반적으로 음성을 packet switching방식으로 전송할 경우 여러가지 잇점이 있다.[5] KORNET에서는 앞으로 음성과 data를 통합처리할 수 있는 통신망 구축을 목표로하여 VC방식을 선택하였다.

국제표준연맹(ISO)에서는 1977년 다양한 computer 망들의 연결을 위해서 physical, data link, network, transport, session, presentation, application layer 등의 7 layer model을 제시했는데,[6] 개발된 KORNET은 국제표준연맹의 7 layer model을 근간으로 하여 구현하였다.

본 논문은 한국 최초로 개발된 packet switching 방식에 의한 computer 통신망 개발에 관한 제 2 부의 논문으로써, 여기에서는 제 1 부의 KORNET의 개요와 NMC 개발에 관한 논문에 이어[7] KORNET의 설계 및 NNP개발에 관해 중점적으로 다루고자 한다.[8],[9] NNP software개발에 관한 자세한 사항은 제 3 부와 제 4 부의 논문에서 중점적으로 다루고 있다.

본 제 2 부 논문은 서론에 이어 제 Ⅱ 장에서는 KORNET의 buffer management 및 flow control과 routing 방법에 대해 기술하고, 제 Ⅲ 장에서는 NNP의 hardware 및 software구조에 대해 기술하며, 제 Ⅳ 장에서는 NNP system의 개발 단계에 대해 간단히 언급한다. 마지막으로 Ⅴ 장에서 결론을 맺는다.

Ⅱ. KORNET의 Resource Management방법

Computer통신망의 성능에 가장 큰 영향을 끼치는 resource management에는 다음과 같은 세가지 형태가 있다.

첫째는 node내의 일정한 용량의 buffer를 각 channel에 대해 어떻게 할당해 주느냐 하는 buffer management, 둘째는 network이 과부하에 의한 체증현상과 throughput 감소 방지를 위해 traffic을 제어하는 flow control, 세째로는 network의 상황에 따라 packet의 path를 결정해 주는 routing이 바로 그것이다.

이들 방법을 어떻게 구현하느냐에 따라 통신망의 전체 성능에 큰 영향을 주는데, 본 장에서는 KORNET에서 사용하고 있는 buffer management, flow control 및 routing의 방법에 대해 설명한다.

1. Buffer Management

Buffer management는 flow control과 밀접한 관계가 있는데, node내의 buffer들에 대한 channel간의 공평하고 효율적인 할당과 deadlock방지를 위해 매우 중요하다.

KORNET에서는 network이나 node내의 다른 module로 부터 들어온 packet을 system내에서 처리하는 동안 data를 한 gueue에서 다른 queue로 옮기는 불필요한 copy를 없애기 위해, 들어온 packet은 centrally managed되는 공용 buffer에다 넣고, 각 process의 input queue에는 buffer의 pointer와 data의 size에 대한 정보만 넣어 전해주게 되는데 그림 1 은 이를 나타내고 있다. Central buffer manager는 각 process의 요구에 따라 module내의 새로운 buffer block의 할당 및 release, 그리고 공용 buffer pool의 status check기능을 담당하고 있다. Buffer block의 할당 방법은 공용 buffer pool을 channel당 상한치까지 완전한 dynamic sharing하여 할당하는 sharing with maximum queues(SMXQ) 방식을 사용하고 있는데, 이는 다른 방법들에 비해 비교적 좋은 성능을 가진다.[10]

개설된 channel의 수를 N이라 하고, 공용 buffer pool의 block의 총 수를 B라 하면, SMXQ방식에 의한 channel당 buffer 할당의 최대치 b_{max}는 complete partition(CP) 방식에 의한 buffer 할당치 B/N와는 다음 식과 같은 관계가 있다.

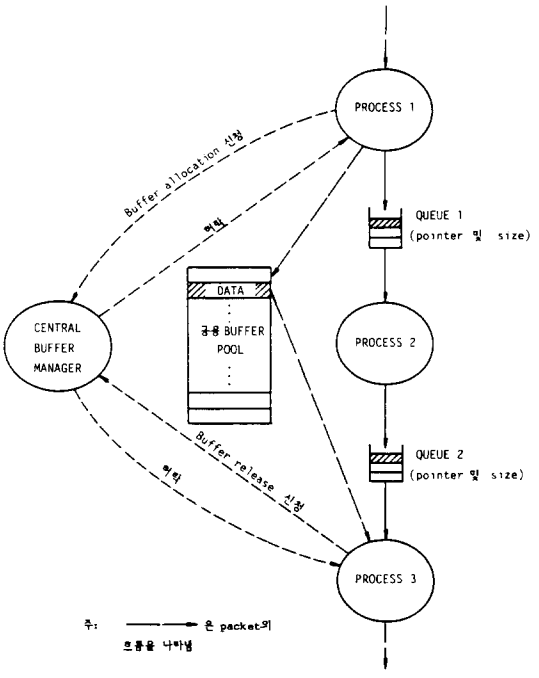

그림 1. KORNET의 buffer management의 예
Fig. 1. Example of KORNET buffer management.

$$b_{max} > B/N \qquad (1)$$

각 channel은 최대 b_{max}까지의 buffer block을 할당받을 수가 있는데, central buffer manager는 아래의 식(2)와 (3)이 만족하는 범위에서 buffer block을 할당해 주게 된다

$$0 \leq n_i \leq b_{max},\ V_i,\ i=0, 1, \cdots, N. \qquad (2)$$

$$\sum_{i=1}^{N} n_i \leq B \qquad (3)$$

여기서, n_i는 i channel이 할당받은 buffer block의 총 수를 나타낸다.

KORNET에서는 이와같은 buffer management 방법으로 buffer를 dynamic sharing함으로써 storage에 대한 이용 효율을 높였다.

2. Flow Control

앞에서 언급했듯이 packet 교환방식은 종래의 circuit 교환방식에 비해 channel과 buffer 및 switching processor를 효율적으로 dynamic sharing한다는 점이 장점이지만, system 처리 용량보다 입력되는 traffic이 많을 때는 체증현상이 생겨 급격히 그 효율성이 떨어지며, packet의 지연이 커지게 된다. 이러한 체증현상에 의한 급격한 통신망의 성능감소를 방지하기 위해 입력되는 traffic량을 제어하는 flow control이 필요하게 되는데, 이 flow control의 주요한 기능으로는 다음과 같은 것이 있다.[3]

- 과부하에 의한 throughput 및 효율성 감소방지.
- Deadlock 방지.
- 여러 사용자들간에 resource의 공평한 할당.
- Network과 접속된 단말기 사이의 speed matching.

그런데, flow control을 하게되면 체증방지에 의한 효율성 증대와, 제한된 resource sharing과 overhead에 의한 효율성 감소의 trade-off가 있다.

KORNET에서의 flow control은 크게 나누어 link layer와 packet layer의 window를 사용한 flow control, output channel queue limit에 의한 flow control 및 PAD의 input buffer limit에 의한 flow control 등이 있다.

Link layer 및 packet layer의 window flow control은 다음의 4가지 형태로 나누어 볼 수 있다.

- 매 information frame이나 data packet의 수신시 마다 acknowledge (ACK) 하는 방법.
- Fixed window flow control이면서 window끝점에서 ACK하는 방법.
- Fixed window flow control이면서 window의 시작점에서 ACK하는 방법.
- Sliding window flow control.

이러한 4가지의 window flow control은 나름대로의 특색이 있으며 channel error rate에 따라 각각의 효율성이 독립적으로 변하는데, 대체로 channel의 이용 효율면에서 볼 때 sliding window flow control 방식이 가장 효율적인 것으로 알려져 있다.[1]

KORNET의 link layer 및 packet layer는 CCITT 권고사항을 따라 window size를 각각 7 및 2로 사용하고 piggyback ACK 방법에 의한 sliding window flow control을 채택하였다. 또한, output channel queue의 상황에 따라 더 이상의 information frame이나 data packet을 처리하여 전송할 수 없을 경우나 새로이 buffer를 할당받을 수 없을 경우는, 각각 송신측에 receive not ready (RNR) frame을 전송하거나 RNR packet을 전송하여 송신 중지를 요청한다.

한편 packet assembly/disassembly (PAD) 에서는 start/stop mode DTE (S/S DTE)로 부터 들어오는 character들을 일정한 크기로 나누어진 input line buffer (ILB)에 차례로 쌓아 packet assembly의 기능을 수행한다. 이때 ILB에 쌓인 block의 수가 일정한 한계치를 넘으면 PAD는 S/S DTE로 X-OFF (DC3)를 전송하여 character의 송신을 멈추게 할 수 있고, 다시 ILB가 사용 가능하게 되면 S/S DTE에게 X-ON (DC1)을 전송하며 계속적인 character 송신을 허락할 수 있다.

3. Routing

Routing이란 computer 통신망의 상태변화에 따라 source node와 destination node 사이에 최소의 비용이 드는 적절한 경로의선택을 말하는데, flow control과 buffer management와 더불어 deadlock 및 체증방지와 channel의 공정한 운영을 위해 매우 중요한 역할을 한다. 일반적으로 routing algorithm은 정확성(correctness), 공정성(fairness), 단순성(simplicity), 안정성(stability), 최적성(optimality) 및 robustness 를 만족하도록 설계하여야 하는데, algorithm 설계시 고려하여야 할 사항들은 다음과 같은 것들이 있다.[8]

- 각 선로의 traffic 측정방법.
- Routing 경로의 제어방법.
- Routing table의 결정 및 경신(update) 방법.
- Topology의 영향.

Routing 경로의 제어 방법에는 크게 고정(fixed)방식과 적응(adaptive)방식으로 구분 할 수 있다. 고정 routing방식은 간단하고 안정성은 있으나, 갑작스러운 traffic 상태의 변화에 대처하기 곤란하고, 공정성 및 최적성이 문제가 될 수 있다. 반면에 적응 routing 방식은 traffic 변화에 쉽게 대처할 수 있으나 제어방법이 복잡하고, 안정성에 영향을 줄 수 있다. 한편 적응 routing 방식은 routing의 제어를 어디서 하느냐에 따라 중앙집중형(centralized) routing, 고립형(isolated) routing, 분산형(distributed) routing, 및 혼합형(hybrid) routing 등으로 구분된다.[2],[12]

KORNET에서는 VC방식으로 packet service를 하고 있기 때문에 routing이 session 단위로 이루어지는데, ARPANET의 방식[12]과 유사한 분산적응형 routing방식을 사용하고 있다.

(1)각 선로에 대한 traffic 측정 model

Network의 각 node는 인접 node간의 선로에 대한 traffic 측정을 위해서 일정한 주기로 delay측정용 packet을 전송하여 해당 선로에 대한 round trip delay를 측정한다. 이렇게 하여 측정된 node i와 node j사이의 선로 L(i, j)에 대한 round trip delay를 d(i, j) 라 하면, d(i, j)는 대략적으로 다음 식과 같이 표현할 수 있다.

$$d(i,j)=\frac{2E[\tilde{\ell}]}{Cp(i,j)}+2(D(i,j)+P)+\frac{\rho(i,j)\cdot E[\tilde{\ell}^2]}{2E[\tilde{\ell}]\cdot Cp(i,j)} \quad (4)$$

여기서, $E[\tilde{\ell}]$은 frame의 평균길이(bits), Cp(i, j)는 선로의 용량(bits/sec), D(i, j)는 선로의 propagation delay, P는 node의 processing delay(sec), $\rho(i,j)$는 선로의 utilization을 나타내고 있는데, $E[\tilde{\ell}^2]/2E[\tilde{\ell}]$ Cp(i, j)는 선로의 remaining service time[13]이 된다. 식(4)에서 propagation delay D(i, j)는 사실상 무시할 수가 있고, $E[\tilde{\ell}]$, $E[\tilde{\ell}^2]$ 및 Cp(i, j)는 상수로 주어지므로, 측정된 round trip delay d(i, j)를 이용하면 식(4)로 부터 선로 L(i, j)에 대한 ulilization $\rho(i,j)$를 계산할 수가 있다.

KORNET에서는 각 선로의 비용 계산을 위한 link weight w(i, j)를 다음과 같이 정의한다.

$$w(i,j)=\rho(i,j)\cdot Fi+Ci \quad (5)$$

여기서, Ci는 node i의 processing capacity에 대한 weight이고, Fi는 node i에 연결된 모든 선로에 대한 mean utilization의 함수이다. 이러한 w(i, j)는 각 경로에 대한 비용 계산을 위한 link weight table을 만드는데 사용된다.

(2)Network topology 및 link weight table

각 node는 network이 개설될 때 topology에 대한 table을 갖게 되는데 그림2와 같은 topology를 갖는 통신망을 예로들어, 1번 node와 2번 node에 대한 topology table을 tree 형태로 나타내면 그림3과 같다. 이와같은 topology table에 따라, 각 node는 주기적으로 새로 측정된 d(i, j)로부터 식

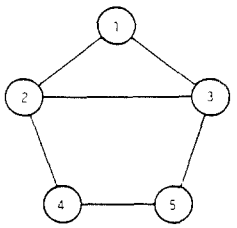

그림 2. 통신망의 topology 예(숫자는 각 node를 나타냄)
Fig. 2. An example of network topology.

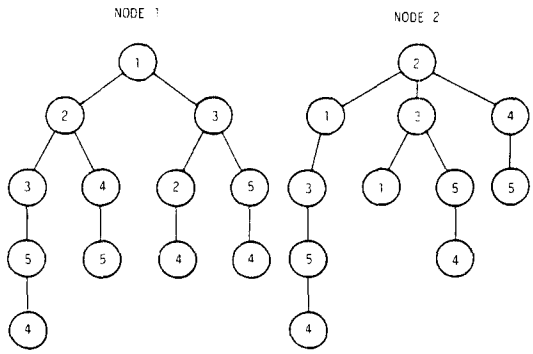

그림 3. Network topology tree
Fig. 3. Network topology tree.

(5)를 이용하여 각 link weight w (i, j)를 계산한다. 새로 계산된 link weight가 예전의 값보다 어느 level 이상 차이가 나면 link weight table을 update 시키고, 모든 node에게 이에 대한 정보를 flooding시킨다.

(3) 최적경로 선정방법 및 routing table

각 node는topology table을 참조하여 destination node까지의 여러 경로에 대한 비용을 계산한 뒤에, 최적 경로를 선정하게 된다. 먼저 각 경로에 대한 비용 Y는 link weight table을 이용하여 식(6)과 같이 계산할 수 있다.

$$Y = (1+\alpha \cdot h) \cdot \sum_{k=0}^{h-1} w_k \qquad (6)$$

여기서, h는 hop수를 나타내며 α는 hop수의 증가에 따른 가중치이고, w_k는 source node와 destination node사이의 k번째 선로에 대한 link weight이다.식(6)에서 α는 network 내에 traffic이 많을 경우 hop수가 큰 경로를 차단하기 위한 가중치인데, 각 node는 식(6)을 이용하여 구해진 여러 경로에 대한 비용 값으로부터 최소 비용을 갖는 2개의 최적경로를 routing table에 update시킨다. 이때, 계산된 경로의 비용이 일정한 값을 넘어서면 해당 경로에 대한 신규 channel 개설을 금지시킨다.

표 1. Node 1에 대한 routing table
Table 1. Routing table of node 1.

Destination node	첫번째 선택경로	Path cost	두번째 선택경로	Path cost
2	2	10	3	22
3	3	10	2	22
4	2	22	3	33
5	3	22	2	33

앞의 예에 대한 모든 선로의 link weight w_k를 10이라 하고 hop수에 대한 가중치 α를 0.1이라두고 node1에 대한 routing table을 구하면 표1과 같다. 여기서 첫번째 선택 경로와 두번째 선택 경로를 두는 것은 node내의 갑작스런 상태 변화에 따른 routing경로 선정에 대한 안정성을 높이기 위함이다. 표1에서 알 수 있듯이, routing table은 source node에서 destination node 사이의 모든 node에 대한 기록이 필요없고 경로상의 다음 node만 지정해 주면된다. 예를들면, 1번 node에서 2번 node 또는 4번 node의 사용자와 channel을 개설하고자 할 때 1번 node는 routing table에 따라 2번 node로만 channel을 개설해 주면된다.

한편, routing table의 update 주기는 통신망 전체의 안정성과 공정성에 큰 영향을 미치기 때문에 신중히 고려하여 결정하여야 하는데, KORNET에서는 실험을 통해 20초를 주기로 잡고 있다.

Ⅲ. Network Node Processor (NNP) 의 구조

1. NNP Hardware의 구조

NNP hardware는 16 bit microprocessor인 MC 68000 CPU와 peripheral device들을 이용하여 data의 실시간(real time) 처리를 위해 multiprocessor system으로 구현하였는데, back plane은 Euro-card 형태인 VMEbus를 사용하였다. 그리고, system의 용량 확장을 용이하게 하기 위해 기능별로 modularity를 살려, 그림4와 같이 master control processing module (MCPM), line processing module (LPM) 및 common memory module (CMM)로 구성하였다.[9]

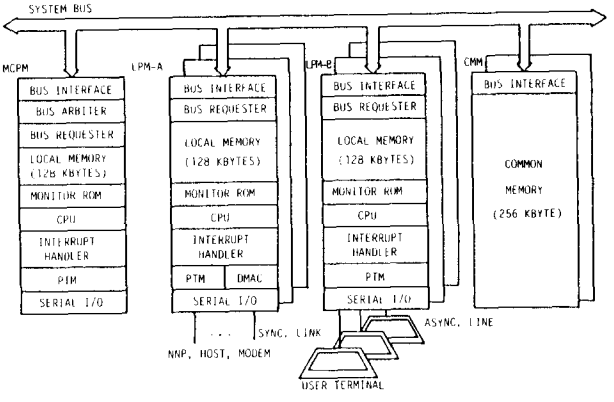

그림 4. NNP hardware의 구조
Fig. 4. Hardware structure of NNP.

이들 각 module 별 특징을 살펴보면 다음과 같다.

(1) MCPM의 hardware

이 module은 MCPM 및 여러 LPM간의 CMM을 이용한 multiprocessing 시의 data 교환을 관장한다. 즉, bus arbiter를 가지고 있어서 system bus 를 daisy chain 형태로 제어하게 되어 있다.

이 밖에 16bit microprocessor인 MC68000 CPU, programmable timer, monitoring을 위한 serial input/output (I/O), physical interface, interrupt handler, 그리고 local memory 등으로 구성되어 있다. Local memory의 용량은 128kbytes로 초기에는 static RAM을 사용하였는데, software 개발이 완료된 후 EPROM으로 쉽게 대체할 수 있도록 설계하였다.

(2) LPM의 hardware

LPM은 packet mode data terminal equipment

(PDTE)나 X.25용 link의 접속을 위한 LPM-A와 S/S DTE의 접속을 위한 LPM-B의 두 종류로 되는데, MCPM과는 달리 memory map 방식으로 여러개의 serial I/O port가 달려 있다. 그리고, LPM-A는 그림 5에서 보듯이 고속의 입출력을 위해 direct memory access controller (DMAC)가 사용된다. 그밖에 LPM module은 bus arbiter가 없는 점 이외에는 MCPM구조와 거의 같다.

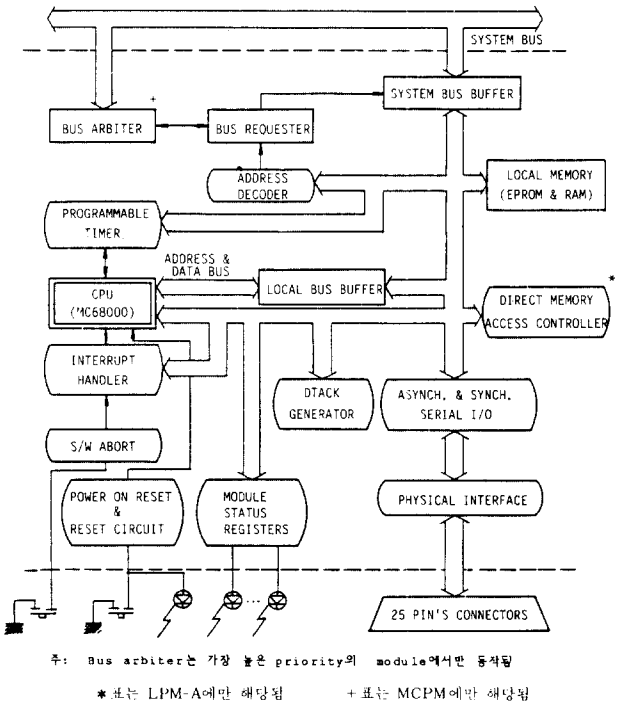

그림 5. MCPM 및 LPM의 block diagram
Fig. 5. Block diagram of MCPM and LPM.

(3) CMM의 hardware

CMM은 각 MCPM 및 LPM들의 module간 정보교환을 위한 공용 memory로서 사용된다. CMM의 memory 용량은 256kbytes로써 64kbytes마다 CMM board의 memory 영역 및 base address가 변경될 수 있으며, bus failure에 의한 data error를 감지할 수 있도록 even parity checking 기능이 첨가되어 있다.

NNP hardware module의 총 확장 가능한 수는 20module로, MCPM이 2 module, LPM이 16module, CMM이 2module인데, 이러한 NNP system의 hardware 특징을 요약하면 표 2와 같다.

2. NNP software의 구조

NNP system은 앞에서 설명하였듯이 각 module의

표 2. NNP system의 hardware 특징
Table 2. Hardware characteristics of NNP system.

CPU	MC 68000
Module 구조	
MCPM	LPM-A, LPM-B, CMM 관장
LPM-A	Synchronous serial I/O
LPM-B	Asynchronous serial I/O
CMM	각 module 간의 정보교환을 위한 공용 memory
전체 address 영역	16 Mbytes
Local memory 영역	128 kbytes
Monitor ROM	16 kbytes (Debugging program)
공용 memory 영역	256 kbytes
총 확장가능 module 수	20 modules
MCPM	2 modules
LPM	16 modules
CMM	2 modules
Multitasking 및 multiprocessing	각 module에는 multitask real time용OS가 load되며, CMM을 통해 module간 정보교환이 된다.
I/O port수/1 LPM	
- Synchronous	56 kbps × 2 ports Low speed × 14 ports
- Asynchronous	Max. speed 19.2 kbps × 16 ports
System bus	VME bus interface

기능과 특성에 따라 MCPM, LPM-A, LPM-B의 3가지 형태의 switching processor와 CMM으로 구성되어 있다. 이들 NNP system에 내장되는 software의 구조는 그림 6과 같은데, module별로 software를 설명하면 다음과 같다.

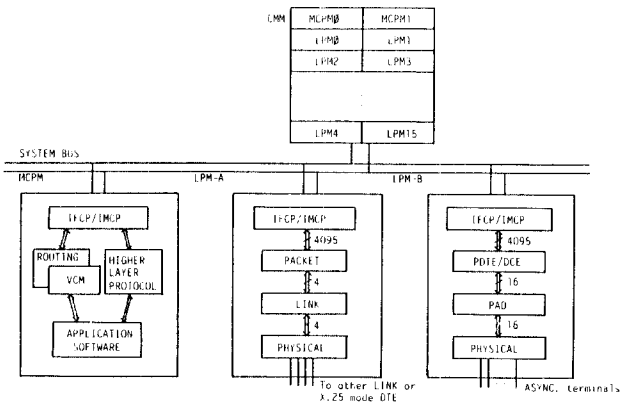

그림 6. NNP의 전체 software 구조
Fig. 6. Overall structure of NNP software.

(1) MCPM의 software

MCPM은 크게 나누어 4가지 routine으로 구분된다. 첫째는 module내의 각 channel의 상태를 조정 및

감독하여 교환 기능을 수행하는 virtual circuit management(VCM), 둘째는 각 선로에 대한 체증(congestion) 및 고장(failure) 상태를 점검하여 routing path의 선택과 감독을 하는 routing management, 셋째는 NMC로 부터 내려온 각종 명령을 수행하기 위한 higher layer protocol과 application software, 마지막으로 각 module 및 function간의 data교환을 위한 interstation communication protocol(ISCP)이 있다.

VCM은 사용자가 통화로의 개설을 신청할 경우, routing management로부터 그때의 선로의 상태에 따라 적절한 routing path를 받아 통화로를 개설해 주며 사용자 선택 기능등을 처리한다. Routing management는 분산적응방식의 routing을 사용하는데 각 선로 상태의 변동을 인접 node에 알려서 routing path의 선택에 참고하도록 하며 VCM에 channel 개설을 위한 자료를 제공한다. Higher layer protocol은 국제표준 연맹의 transport layer와 session layer, application layer에 해당하는데, NMC와 통화로를 개설하여 명령을 처리하고 응답하는데 사용되고 있다.

(2) LPM-A의 software

LPM-A에는 CCITT 권고사항 X.25[14]을 따르는 packet layer와 link layer가 있으며, module 및 function간의 data 교환을 위한 ISCP 기능을 내장하고 있다. 여기서, LPM-A는 선로를 통해 PDTE와 접속될 경우는 data circuit terminating equipment(DCE)로 동작되며, 다른 NNP system과 접속될 경우는 data switching equipment(DSE)로 동작된다. 외부와의 접속을 위한 physical layer는 CCITT 권고사항 X.21bis 규정을 만족하는 EIA RS-232C를 따랐다.[15]

Link layer의 access 과정으로는 크게 보아 asynchronous response mode(ARM)로 동작하는 link access protocol(LAP)방식과 asynchronous balanced mode(ABM)로 동작하는 balanced mode link access protocol(LAPB) 방식이 있는데, KORNET에서는 추세에 따라 link layer protocol을 LAPB방식에 근간을 두어 설계하였다. Link layer의 기본적 기능으로는 flag sequence를 응용한 synchronization, DTE와 DCE를 구별해 주는 addressing, 2 byte의 frame check sequence(FCS)를 응용한 error control및 1 byte의 control field를 사용한 flow control 등이 있다.

Packet layer는 subnetwork 뿐만 아니라 PDTE와 subnetwork 사이에도 VC방식으로 packet service를 하고 있는데, logical channel number를 이용한 multiplexing과, 수신 및 송신 sequence number를 사용한 flow control 및 error control을 하고 있다.

그림 7은 KORNET에서의 일반적인 packet 및 frame

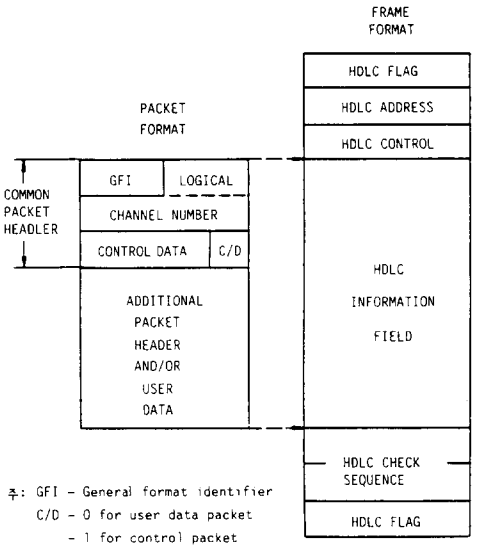

그림 7. KORNET의 일반적인 packet 및 frame formats
Fig. 7. General packet and frame formats of KORNET.

의 format을 나타내고 있다.

(3) LPM-B의 software

LPM-B는 KORNET에 S/S DTE를 접속하기 위해 구현된 module이다. 크게 나누어, CCITT 권고사항 X.3, X.28, X.29를 따르는 PAD routine, node내의 packet layer와의 접속을 위한 packet DTE/DCE, 각 module 및 function간의 data 교환을 위한 ISCP, 그리고 physical layer와의 접속을 위한 physical layer interface 부분으로 되어있다.

PAD routine은 각종의 S/S DTE를 접속하기 위한 18가지의 PAD parameter를 가지고 있는데, 그 기본적인 기능은 S/S DTE로부터 들어온 character들을 packet으로 assemble하는 기능과 network을 통해 전송되어온 data packet을 character로 disassemble하는 기능 및 통화로의 개설과 단절을 요구하는 기능 등이 있다.

Packet DTE/DCE routine은 LPM-A의 packet layer의 counter part로서 PAD에 연결된 S/S DTE 입장에서 보았을 때는 DCE형태로 동작하고, subnetwork에서 보았을 때는 PDTE처럼 동작하는데 LPM-A의 packet layer보다는 상당히 간략화된 형태로 구현되어 있다.

(4) CMM을 이용한 module간의 data교환

CMM은 MCPM 및 여러 LPM사이에 data 교환을 위해 사용하는 공유 영역인데, 그림 6에서 보듯이 module별로 memory가 할당되어 있다. 이러한 CMM을 이용한 module간의 data교환은 각 module에 내장

되어 있는 ISCP중 intermodule communication protocol(IMCP)에 의해 이루어지는데, 일단 자기 module로 옮겨진 data는 interfunction communication protocol(IFCP)에 의해 해당 function까지 data가 전송된다.

(5) NNP software의 상호관계

NNP에는 LPM-A를 통해 다른 NNP나 PDTE가 접속될 수 있고, LPM-B를 통해 S/S DTE가 접속될 수 있는데, 그림 8 과 9에는 서로 다른 두개의 NNP에 접속된 PDTE사이의 통신과 S/S DTE사이의 통신을 예로들어 NNP software의 상호 관계를 나타내었다.

PDTE에는 NNP의 software에 대응하여 physical, link, packet layer 뿐만 아니라 higher layer protocol도 내장되어 있는데, 이 중에는 virtual terminal 기능을 위한 virtual terminal agent(VTA), file transfer 기능을 위한 file transfer agent(FTA) 등이 있다. PDTE에 연결된 단말기를 통해 사용자가 통화로 개설을 요구하면 PDTE는 새로운 logical channel을 할당하여 call request packet을 인접 NNP로 전송한다. 선로를 통해 전송되어 온 call request packet은 LPM-A의 physical, link, packet layer를 거쳐 ISCP에 의해 MCPM의 VCM으로 전해지고, VCM에서 다른 선로에 대한 logical channel을 새로이 할당받아 다시 다른 NNP로 전송한다. 이와같은 방법으로 call request packet이 remote PDTE에 도착하면, remote PDTE는 반대경로를 통해 call confirmation packet을 전송함으로써 통화로 개설이 완료된다. 두 PDTE 사이의 data송수신은 개설된 통화로를 따라 이루어지는데, call request packet과는 달리 NNP의 VCM을 거치지 않고 physical, link, packet layer를 통해서 이루어진다. 통화로 단절을 위한 clear request packet은 NNP의 VCM으로 전해져 통화로 개설시와는 반대로 해당 logical channel을 취소하고, clear request packet을 전송해온 PDTE나 NNP에게 local significance로 clear confirmation packet을 전송함으로써 완료된다.

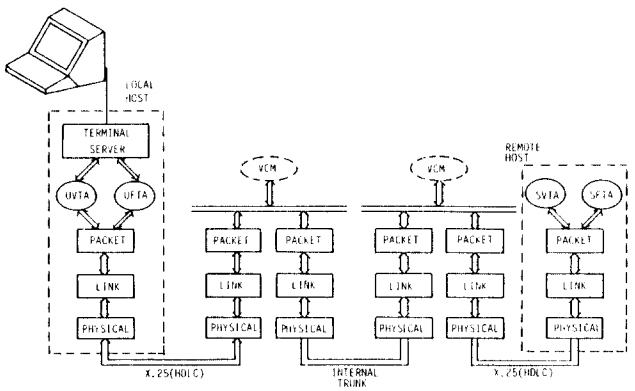

그림 8. 두 PDTE 사이의 통신의 예
Fig. 8. An example of communication between two PDTE's.

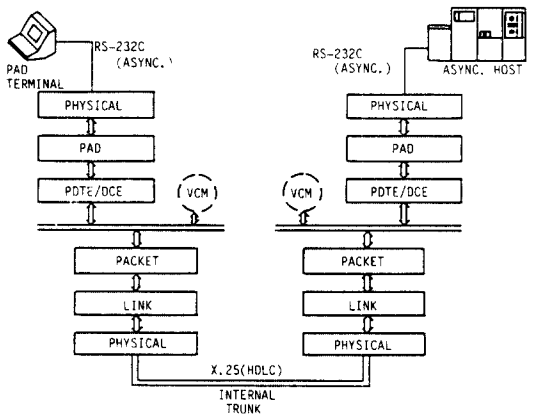

그림 9. 두 S/S DTE 사이의 통신의 예
Fig. 9. An example of communication between two S/S DTE's.

한편 S/S DTE는 NNP의 LPM-B를 통해 접속되는데, 연결되는 단말기의 특성에 따라 PAD parameter들의 값을 적절히 선택하여야 한다. 가입자가 S/S DTE를 통해 통화로 개설을 요구하면 PAD에서 만들어진 call request packet이 packet DTE/DCE에서 새로운 logical channel을 할당받아 VCM으로 전해지는데, 통화로 개설과 단절 및 data 송·수신 과정은 PDTE의 경우와 마찬가지이다.

Ⅳ. Network Node Processor (NNP)의 개발 단계

KORNET은 1차적으로 3개의 NNP와 하나의 NMC로 구성하였는데, 본 장에서는 KORNET의 개발 단계중 NNP system의 개발 과정에 대해 간략히 언급하겠다.

NNP의 software 및 hardware의 개발을 위해서 microprocessor development system (MDS) 및 emulator, logic analyzer, protocol tester 등을 사용했다. 이 밖에 coding의 편의를 위해 MV/8000 computer등을 사용하였는데 그림10은 이를 나타내고 있다.

Software의 초기 설계 단계에서는 CCITT에서 권고하고 있는 specification and description language (SDL)을 사용하여 각 기능을 기술하고 이에 대한 보조자료로 table 및 software structure를 그려서 보완하였다.[16] 일단 SDL이 완성되면 coding이 바로 가능하지만 software design 단계에서 최대한 error를

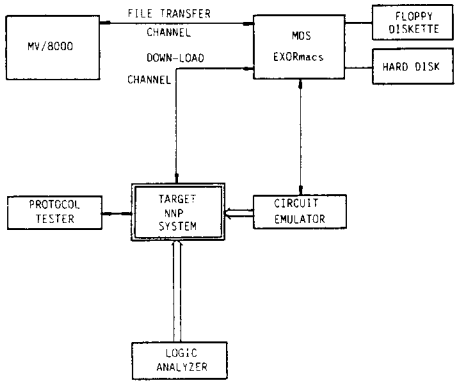

그림10. NNP system 개발 환경
Fig. 10. NNP system developing environment.

검출하고 document를 하기 위해서 program design language (PDL)을 사용하여 coding을 위한 최종documentation을 행하였다.[17] Coding에서 사용된 언어는 NMC의 경우는 PL/I을 이용하였고, NNP는 대부분 C-language를 사용하였으나, 실시간 처리가 요구되는 부분에 대해서는 assembly 언어를 사용하였다.

이와 같은 방법으로 coding이 완료되면 protocol verification을 위해 testing 및 debugging 단계로 들어가는데, 순차적으로는 error를 검출해 나가기 위해 다시 몇 단계로 나누었다. 먼저, MDS에서 각 module 별로 integration test를 거친 다음, 실제 target system에서의 상황을 simulation하는 real time simulation을 행하였다. MDS에서의 test가 끝나면 down loading을 통해 target system인 NNP에서 emulation testing을 하는데, MDS에서 보다 debugging tool의 제한으로 많은 어려움이 있었다. Target system에서의 emulation test가 끝난 후에 3개의 NNP와 하나의 NMC를 연결하여 통신망을 구성한 뒤에 NMC기능 및 NNP 기능에 대한 최종 test를 하였다. 이때는 주로 통신망에 PDTE 및 S/S DTE를 접속하여 virtual terminal agent (VTA) 및 file transfer agent (FTA) 기능에 대한 test를 완료하였다.

Test를 진행함에 있어서 무엇보다 중요한 것은 적절한 check point와 test input의 선정, 그리고 test scenario의 구상이라 할 수 있다. Test를 통해 발생했던 error들을 보면 주로 layer간의 protocol mismatch에 의한 것인데 message queue의 overflow 및 update 잘못, task간의 synchronization, buffer의 allocation 및 deallocation 등이 중요한 문제였다.

위와 같은 error를 해결하기 위해 protocol tester, logic analyzer, bus state analyzer, debugger, terminal display를 이용한 monitor 및 monitor program 등을 주로 이용하였다.

Software의 design 단계에서는 top-down approach를 사용하고, test 및 debugging 단계에서는 bottom-up approach를 사용하여 NNP 및 NMC의 software를 개발 완료하였다. 이와 같은 접근 방법은 커다란 통신 system의 개발에 있어서는 매우 좋은 방법이라 할 수 있다.

V. 結 論

본 논문에서는 packet 교환 방식을 사용한 computer 통신망인 KORNET의 설계와 NNP개발에 대하여 기술하였다. KORNET의 1차형태는 3개의 node (NNP)와, 통신망의 운영과 관리를 맡고 있는 하나의 NMC로 구성하였는데, NNP는 data의 실시간 처리를 위해 16bit microprocessor를 이용하여 multiprocessor system으로 구현하였고, NMC는 MV/8000 computer를 이용하여 개발하였다. KORNET은 앞으로 음성까지 전송할 수 있는 통합 통신망의 구축을 목표로 하여 packet service 방식으로 VC방식을 채택하였고, 통신망의 node나 선로의 상태 변화에 쉽게 대처할 수 있도록 분산적응방식의 routing을 사용하였다. NNP system은 내장되는 software에 따라 3가지 형태의 module로 구성되는데, X.25의 link layer와 packet layer 기능이 구현되어 있어서 NNP나 PDTE가 연결되는 LPM-A, S/S DTE의 접속을 위해 PAD 기능이 구현된 LPM-B, virtual call에 대한 routing과 switching을 담당하고 node의 제반 상황을 감독·통제하는 MCPM으로 되어 있다. 이처럼 NNP system은 modularity를 고려하여 구성하였기 때문에 용량 확장이 용이하며, CCITT 권고사항 X.25, X.3, X.28 그리고 X.29 등을 따라 구현했기 때문에 이를 만족하는 다른 computer망이나 PDTE와 접속이 가능하도록 되어있다.

參 考 文 獻

[1] G.V. Bochman, *Concepts for Distributed Systems Design,* Springer-Verlag, 1983.
[2] A.S. Tanenbaum, *Computer Networks,* Prentice-Hall, 1981.
[3] L. Kleinrock et al., "Flow control : A comparative survey," *IEEE Trans. Commun.,* vol. COM-28, no. 4, pp. 553-574, April 1980.
[4] B.W. Lampson et al., *Distributed Systems-Architecture and Implementation*. Springer-Verlag, 1981.

[5] C.J. Weinstein et al., "Experience with speech communication in packet networks," *IEEE J. Select. Area. Commun.*, vol. SAC-1, no. 6, pp. 963-980, Dec. 1983.

[6] H. Zimmermann, "OSI Reference Model-The ISO model of architecture for Open System Interconnection,"*IEEE Commun.*, vol. COM-28, no. 4, pp. 425-432, April 1980.

[7] 은종관외, "Packet Switching에 의한 공중 Computer 통신망 개발 연구-제 1 부 : KORNET의개요와 Network Management Center(NMC) 의 개발", 한국과학기술원, 1985.

[8] 은종관외, Packet Switching에 의한 Computer 통신망 개발-KORNET 설계, Routing및 Flow 제어 연구, 한국과학기술원 통신공학연구실, 1 차년도 최종보고서, CRLP-8301, 1983.

[9] 은종관외Packet Switching에 의한 Computer 통신망 개발-NNP Software 및 Hardware, 한국과학기술원 통신공학연구실, 2 차년도 최종보고서, CRL-P-8401, 1984.

[10] F. Kamoun and L. Kleinrock,"Analysis of shared finite storage in computer network node environment under general traffic conditions," *IEEE Trans. Commun.*, vol. COM-28, pp. 992-1003, July 1980.

[11] M. Schwartz, "Performance analysis of the SNA virtual route pacing control," *IEEE Trans. Commun.*, vol. COM-30, no. 1, Jan. 1982.

[12] M. Schwartz et al., "Routing techniques used in computer communication networks," *IEEE Trans. Commun.*, vol. COM-28, no. 4, pp. 539-552, April 1980.

[13] L. Kleinrock, *Queueing Systems Volume I : Theory*, Wieley-Interscience, 1975.

[14] CCITT Recommendations X.1-X.29, Yellow Book, vol. VIII.2, Geneva, 1980.

[15] EIA Standard RS-232C, *Interface between data terminal equipment and data communication equipment employing serial binary data interchange*, Aug. 1969.

[16] A. Rockstron et al., "SDL-CCITT specification and description language," *IEEE Trans. Commun.*, vol. COM-30, no. 6, pp. 1310-1318, June 1982.

[17] M.V. Zelkowitz et al., *Principles of Software Engineering and Design*, Prentice-Hall, 1979.