KORNET - THE LATEST PUBLIC PACKET-SWITCHED NETWORK

o
C. K. Un and D. H. Cho

Communications Research Laboratory
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology
P.O. Box 150, Chongyangni
Seoul, Korea

ABSTRACT

This paper describes the development of the KORNET that may be regarded as the latest public packet-switched computer communication network. The KORNET project included the development of the network management center (NMC), a network node processor (NNP) and a network concentrator. For the KORNET we use the virtual circuit (VC) method, a distributed adaptive routing algorithm, and a dynamic buffer management algorithm. The NMC acts as a nerve center of the network, performing such functions as network monitoring, subscriber and network management and routing management, etc. As for the NNP and NC hardwares, we have implemented them with the 16-bit multitask/multiprocessor technology using MC68000 microprocessors. Softwares have been developed using C language except for some parts where assembly language is required for real time processing. All the network protocols we have developed comply completely with the latest CCITT recommendations including X.25, X.3, X.28 and X.29.

I. INTRODUCTION

Since the ARPANET came into being in 1969, several public packet-switched communication networks, such as Telenet, Tymnet, Transpac and Datapac, have been developed in the past 15 years. As the data communication users increase rapidly, the role of these networks is becoming as important as the circuit-switched voice networks.

The purpose of this paper is to present the KORNET that may be regarded as the latest public packet-switched communication network. This network has been developed during the last four years (1981-1985). The KORNET presently serves data signal only, but is expected to accommodate voice as well as data in the near future. The network has been implemented using the virtual circuit approach [1], and follows completely all the latest protocols recommended by the CCITT including X.25, X.3, X.28 and X.29 [2]. The KORNET project involved the development of all the network elements, that is, the network management center (NMC), a network node processor (NNP) and a network concentrator (NC). These elements have been developed using the latest technologies and protocols. The NNP and NC hardwares have been implemented using MC68000 microprocessors, and all softwares have been written in C language except for some parts which require assembly language for real time processing. In this paper we present details of the development of the NMC and NNP.

Following this introduction, in Section II we first describe the resource management method used in the KORNET. In Section III we discuss the development of the NMC which monitors and controls the network. In Section IV we consider the development of the NNP hardware which does packet switching, and in Section V the implementation of the NNP softwares including X.25 and packet assembly/disassembly (PAD) protocols. Finally, we make conclusions in Section VI.

II. RESOURCE MANAGEMENT IN THE KORNET

For the resource management that is critical for sat~~isfa~~ctory performance of a computer communication network, one must consider the following three items: NNP buffer management, network flow control and routing. These are now described.

(1) Buffer Management

Buffer management which has a close relationship with flow control is important for the prevention of deadlock and for the fair allocation and efficient usage of a buffer among various channels.

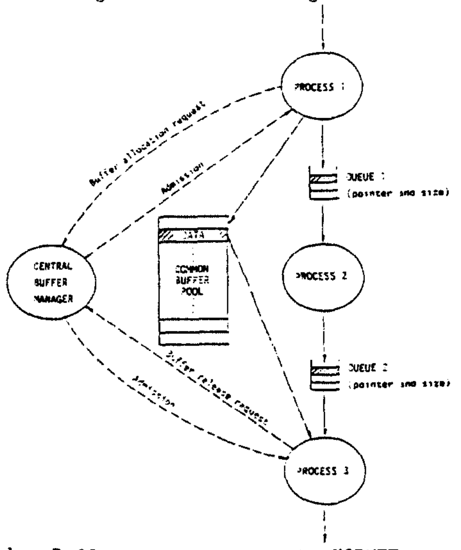

Fig. 1 Buffer management in the KORNET.

A simplified diagram of buffer management in the KORNET is shown in Fig. 1. The common buffer which is centrally managed stores packets from the network or from a module within the NNP, and the input queue of each processor stores only the information regarding the buffer point and the data size. The functions of the central buffer manager are to allocate and release a block of buffer according to the request of each processor, and to check the status of the common buffer pool. In the KORNET the buffer is shared dynamically, thereby improving the efficiency of buffer usage. Buffer blocks are allocated by the method of sharing maximum queues (SMXQ) [3]. Let N be the number of established channels and B be the total number of blocks of the common buffer. When the SMXQ method is used, the following inequality holds between the maximum number of allocated buffers per channel and the number of buffers, B/N, allocated according to the complete partition method:

$$b_{max} > \frac{B}{N} .$$

Hence, the number of buffer blocks n_i allocated to the i_{th} channel is constrained by the following inequalities:

$$0 \leq n_i \leq b_{max}$$

and

$$\sum_{i=1}^{W} n_i \leq B.$$

(2) Flow Control

It is well known that when the amount of data traffic is larger than the system can handle, congestion occurs. In this case the network efficiency would become degraded abruptly with unacceptable packet delay. Therefore, some form of flow control is normally done to avoid such phenomena [4].

In the KORNET, flow control is done in the hop and network access levels, and may be divided into the following three catagories: Window flow control in the link and packet layers, flow control limiting the output channel queue, and flow control by limiting the input buffer of the PAD. Among various window control schemes, we have chosen for the link and packet layers the sliding window flow control scheme based on the piggybacked ACK method which is known to be most efficient in channel utilization [5]. The window sizes of the link and packet layers are 7 and 2, respectively, which conform to the CCITT recommendation. Also, in the case that no more information or data packets can be processed or no additional new buffer is allocated, interruption of transmission is requested by sending a receive not ready (RNR) frame or RNR packet to the transmitter.

In the PAD, characters from the start/stop (s/s) data terminal equipment (DTE) are stored in order in the input line buffer (ILB). If the number of blocks stored in the ILB exceeds some limit, the PAD transmits the X-OFF signal (DC3) to the s/s DTE so that character transmission is temporarily halted. When the ILB is available again, it transmits the X-ON (DC1), thereby allowing character transmission.

(3) Routing

Routing plays an important role together with flow control and buffer management for prevention of deadlock and congestion, and for a fair utilization of communication channels. In general, a routing algorithm must be designed considering accuracy, fairness, simplicity, stability, optimality and robustness. One can think of two routing methods; fixed routing and adaptive routing [6]. In the KORNET, a distributed adaptive routing method that is similar to the scheme used in the ARPANET is used in the transport level since the packet service is based on the virtual circuit method.

III. NETWORK MANAGEMENT CENTER

The NMC hardware has been implemented using the 32 bit MV/8000 superminicomputer of Data General (DG), Inc. NMC Softwares, such as operator interface, primary/secondary (P/S), session layer, packet layer and link layer, have been developed in C language except for some part of link layer which requires assembly language for real time processing. For implementation of the physical layer, an intelligent synchronous controller (ISC) board supplied by Data General has been used. Since packet and link layer protocols of the NMC are the same as those of the NNP, only higher layer softwares above the session layer will be explained in this section.

(1) Operator Interface

The operator uses the network command language (NCL) based on CCITT Z.311-Z.341 recommendations to check the network status and to manage the network elements [7]. Three-stage jobs, i.e., command acquisition, command execution and response display, are done in the operator interface part. Table I shows the main commands used in the KORNET. The response to a command is transferred to the operator terminal from the network.

Table I. Major NCL's used in KORNET

Command	NCL command	Content
Session	SE-BGN SE-MOD SE-PVC SE-MSG SE-END	Session open Session modification PVC session open Message exchange between operator sessions Session end
Subscriber management	SA-PVC SD-PVC SM-PVC	PVC creation PVC deletion PVC modification
Routing management	RO-TBL RO-MOD	Routing table collection Routing table modification
Network monitoring	SQ-LNK SQ-PVC SQ-PVA SQ-LCH	Link status monitoring PVC status monitoring Monitoring for the number of PVC's opened Logical channel status monitoring
Network management	TC-CLK TC-DLK	Link connection Logical link disconnection
Other services	HELP COMMAND NCL	Usage of command Record file examination NCL explanation

(2) Primary/Secondary (P/S) Function

When commands and parameters are sent to the P/S queue from the execution controller of the operator interface part, classification for commands is made. Then, a process, such as subscriber management, network management, network monitoring or routing management, is executed, and it is monitored by the communication process monitor. Depending on the command, it is decided to open the session, have the data transfer state, or only access to a table. These commands and parameters are sent to the session layer. Fig. 2 shows the structure of the P/S function. If the response to a command is received, the command and the sequence number are stored in the memory of P/S function, and parameters and the response are recorded in the file system. Typical responses are execution error, time out, and normal response.

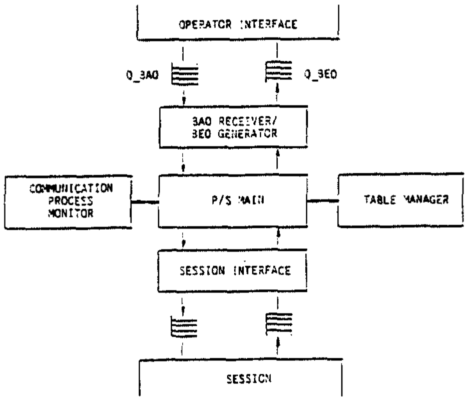

Fig. 2 Software structure of primary/secondary function.

The messages generated periodically or aperiodically in the NMC are sent to the P/S function of an NNP as shown in Fig. 3. When each parameter of message is determined in the P/S of the NMC, the session to communicate with an NNP is open. The information for opening a session is transferred using the internal message. The P/S function of the NNP classifies the received messages from the NMC, and executes a proper action. Then, a logical channel and a routing information from the network node process manager (NNPM) as well as initialized parameters are utilized. Thereafter, the corresponding responses to the command are sent to the P/S function of the NMC.

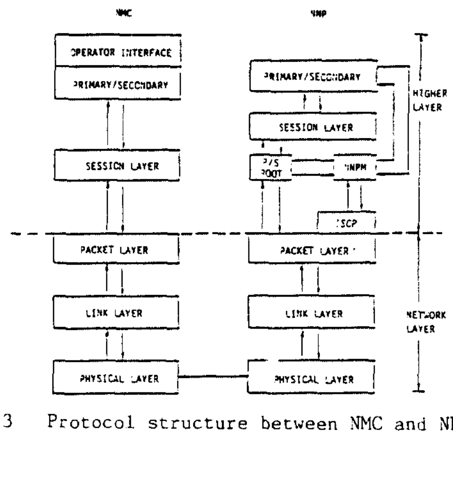

Fig. 3 Protocol structure between NMC and NNP.

(3) Session Layer

The session layer has been implemented based on the session layer protocol proposed by Bell Laboratories as shown in Fig. 4. Four external queues and parameter tables are used to exchange information among layers. When the information is sent to another layer, the related parameters are transferred using an interface table. The data to be sent to the NNP is read into the data buffer of higher layer, and sent to the internal queue (Q-SD). The data in the Q-SD is transferred through the transport service to the NNP after a session header is attached. On the other hand, the received data from the NNP is read into the queue Q-SU from the packet level buffer. If all data messages arrive, the received data messages are transferred to the higher layer.

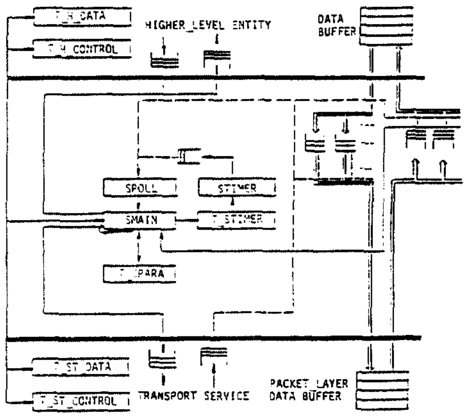

Fig. 4 Software structure of session layer.
(═ data flow, ── information flow, --- polling sequence)

SPOLL executes polling of input queues. If SPOLL detects the events, it informs SMAIN so as to execute a proper action. The creation and deletion of a new logical channel, status change and various parameters are recorded in the T_STIMER table. Also, if time-out occurs, STIMER informs SMAIN of time-out through Q_STIM and SPOLL.

(4) Integration and Test of NMC

To test the NMC, we have followed the following four steps; module test, module integration test, local loopback test, and total integration test after connecting the NMC with NNP's. Each task coded in PL/I language or assembly language has been debugged using various facilities of the MV/8000 computer, and tested by monitoring various queues and tables. Also, module integration and local loopback test have been confirmed by examining the output pattern for various input sequences and by monitoring the contents of queues and tables. At the stage of emulation test, various protocol testers, such as TEK834 and CHAMELEON II, have been used. In this case, the protocol tester and the MV/8000 computer were operated in the DCE and DTE modes, respectively.

The functional operations of the NMC, such as session opening, call setup, data transfer and call clear, were checked according to a test scenario. Also, the functional check for the NNP system was done using the same test scenario.

After the total integration of the NMC and NNP's, we have tested whether the NMC properly sends commands to the network elements such as the NNP, and receives correctly corresponding responses from the network. Moreover, we have tested network application utilities such as the virtual terminal agent and the file transfer agent through the KORNET, and confirmed their operation to be normal.

IV. IMPLEMENTATION OF THE NNP HARDWARE

The NNP hardware has been implemented as a multiprocessor/multitask system for real time processing of data signals using MC68000 16-bit microprocessors and peripheral devices. It has the VME bus structure, and consists of a master control processing module (MCPM), common memory modules (CMM's), and line processing modules (LPM's). In the design of the NNP, a particular emphasis was placed on the modularity for easy expansion of the system capacity. The structure of the NNP hardware is shown in Fig. 5.

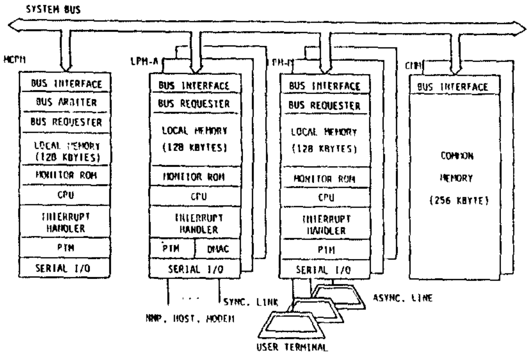

Fig. 5 Hardware structure of NNP.

(1) MCPM

The MCPM governs data exchange between the MCPM and LPM's for multiprocessing through the use of the CMM. It has a bus arbiter and controls the system bus in the form of a daisy chain. A block diagram of the MCPM and LPM is shown in Fig. 6. As seen in the figure, the MCPM consists of an MC68000 CPU, a programmable timer, a serial I/O, a physical interface, an interrupt handler, a local memory of 128 kbytes, etc.

(2) LPM

. The LPM is divided into LPM-A and LPM-B. The former is used for connection with packet-mode data terminal equipments (PDTE's) and X.25 links, and the later for connection with s/s DTE's. It has many serial I/O ports using the memory map method. As seen in Fig. 6, the structure of the LPM-A is almost the same as the MCPM except that it has a direct memory access controller (DMAC), but has no bus arbiter.

(3) CMM

The CMM is used as a common memory for data exchange between the MCPM and LPM's. The size of the CMM is 256 kbytes, and may be expanded. It has the capability of even parity checking, thereby enabling to detect data errors resulting from bus failures. The features of the NNP hardware developed for the KORNET are summarized in Table II.

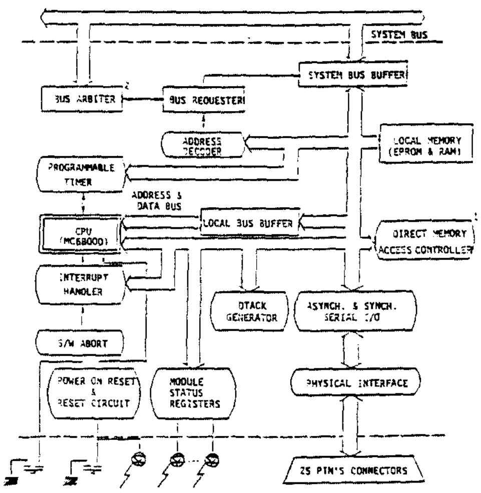

Fig. 6 Block diagram of MCPM and LPM.
Note: 1. The DMAC is included only in the LPM-A module.
2. The bus arbiter is included only in the MCPM module.

Table II. Hardware characteristics of NNP system.

1. CPU	MC68000
2. Module structure	
. MCPM	Control of LPM-A, LPM-B and CMM
. LPM-A	Synchronous serial I/O
. LPM-B	Asynchronous serial I/O
. CMM	Common memory for information exchange among modules
3. Memory size	
. Total address size	16 Mbytes
. Local memory size	128 Kbytes
. Monitor ROM	16 Kbytes (Debugging program)
. Common memory size	256 Kbytes
4. Total module	20 modules
. MCPM	2 modules
. LPM	16 modules
. CMM	2 modules
5. Operating system	
. Multitasking	For each module, multitask real time operating system is included
. Multiprocessing	Information is exchanged using CMM
6. I/O port/1 LPM	
. Synchronous	56 Kbps x 2 ports Low speed x 14 ports
. Asynchronous	Max. speed 19.2 Kbps x 16 ports
7. System bus	VME bus interface

V. IMPLEMENTATION OF NNP SOFTWARES

NNP softwares have been developed using C language except for some parts where MC68000 assembly language is required for real time processing. These softwares have been developed systematically using the specification and description language (SDL) and the program design language (PDL). The overall software structure of the NNP is shown in Fig. 7. The software of each module is now described.

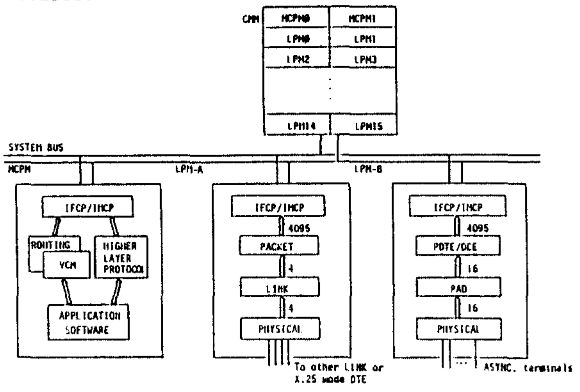

Fig. 7 Overall software structure of NNP.

(1) MCPM Software

The structure of the MCPM software is shown in Fig. 8. It may be divided largely into four routines. First, the virtual circuit management (VCM) routine has the switching and supervisory functions. It also gives necessary commands to LPM-A and LPM-B. When a user requests a communication channel, the VCM opens a channel based on the routing path information from the routing management. Second, the routing management routine detects congestion and failures of each path, and provides the routing path information to the VCM. It uses an adaptive distribution algorithm. It monitors channel states, and informs the neighboring NNP of it to help select a routing path. Third, the higher layer protocols including application softwares carry out and respond to various commands received from the NMC. These protocols correspond to the transport, session and application layer protocols of the CCITT. Lastly,

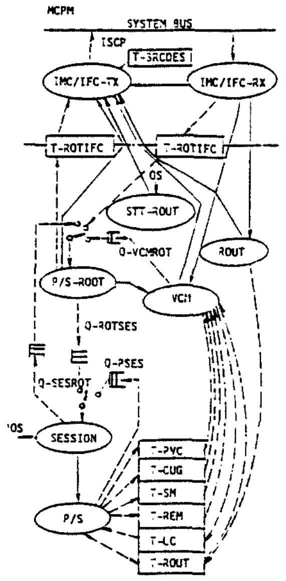

Fig. 8 Overall software structure of MCPM.

there is the interstation communication protocol (ISCP) which is composed of the intermodule communication protocol (IMCP) and the interfunction communication protocol (IFCP). The ISCP is used for data exchange among various modules and functions.

(2) LPM-A Software

The software structure of the LPM-A which services the X.25 protocol is shown in Fig. 9. It consists of PKT-MAIN for the packet layer protocol, ANAL-RCV and FRM-MAIN for the link layer protocol, and the high-level data link control (HDLC) receiver and transmitter modules to interface with the physical layer. In addition, it has the ISCP for communication with other hardware boards through the system bus.

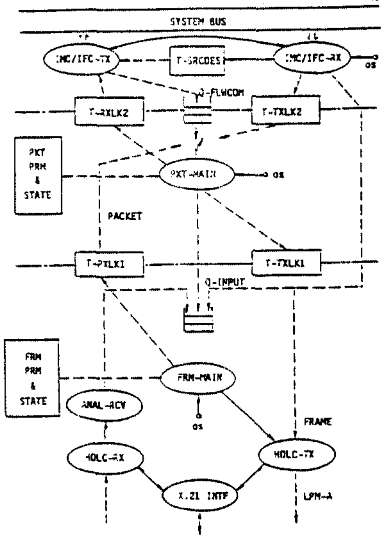

Fig. 9 Software structure of X.25 protocol.

The packet layer provides the packetization services. It decides the packet format, multiplexes logical channels, and does flow and error control using the packet sequence number. Also, it provides optional user facilities, such as a closed user group and a fast select function.

The basic functions of the link layer include synchronization utilizing the flag sequence, addressing that discriminates a DTE from a DCE, error control using the frame check sequence (FCS) of 2 bytes, and flow control using the control field of 1 byte. There are two methods of accessing to the link layer; the link access protocol (LAP) method based on the asynchronous response mode (ARM) and the balanced mode link access protocol (LAPB) based on the asynchronous balanced mode. In the KORNET the link layer protocol has been designed based on the LAPB method.

In addition, the physical layer follows the EIA RS-232C which satisfies the CCITT recommendation X.21 bis.

(3) LPM-B Software

The software structure of the LPM-B is shown in Fig. 10. It may be divided into the following four routines: The PAD routine that follows the CCITT recommendations X.3, X.28, and X.29, the pack-

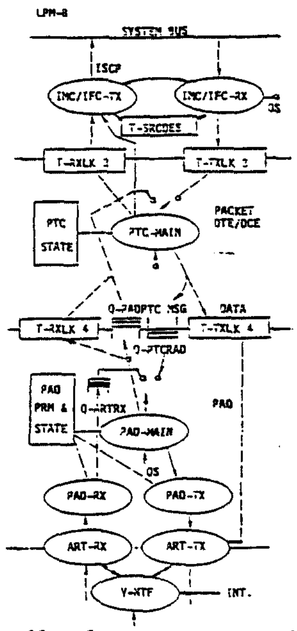

Fig. 10 Overall software structure of LPMB.

et DTE/DCE routine for interface with the packet layer within the node, the ISCP, and the physical layer interface. The PAD routine has 18 parameters for connection with the s/s DTE. Its basic functions include assembling packets using characters received from an s/s DTE, disassembling packets from the network into characters, and setting up and disconnecting channels. The packet DTE/DCE routine which may be regarded as the counterpart of the LPM-A packet layer has been implemented in a simpler form than the LPM-A packet layer.

(4) Interrelationship of NNP Softwares

An NNP is connected to another NNP or PDTE through LPM-A, and to s/s DTE's through LPM-B. The PDTE has physical, link, packet and higher layer protocols including the virtual terminal agent and the file transfer agent. These are counterparts to those stored in the NNP.

The interrelationship of the NNP softwares may be explained using an example of communication between two PDTE's that are connected to their respective NNP's. When a user attached to a PDTE requests a channel, the PDTE is assigned a new logical channel, and transmits a call request packet to a neighboring NNP. This call request packet is transferred to the VCM of the MCPM through the LPM-A by the ISCP. At the VCM, another logical channel is assigned for transmission to another NNP. When the call request packet is received by a remote PDTE, it transmits a call confirmation packet on the reverse route, thereby setting up a communication channel. Communication between the two PDTE's is done through this channel without passing through the VCM of the NNP. To disconnect the channel, a clear request packet is sent to the VCM of an NNP, and the corresponding logical channel is cancelled. Then, a clear confirmation packet is transferred to the PDTE or NNP which sends a clear request packet.

An s/s DTE is connected to the LPM-B of the NNP. The PAD parameter values must be selected properly according to the DTE being connected. When a call is requested through an s/s DTE, a call request packet is sent to the VCM through a logical channel. The procedures of setting-up, disconnection and data transmission are the same as for the case of the PDTE.

VI. CONCLUSION

We have presented the development of a public packet-switched computer network named the KORNET. In its development we have used the latest software and hardware technologies, and the latest protocols recommended by the CCITT. In this regard, the KORNET may be regarded as the most up-to-date public data network at the present time. Also, in the design of network softwares and hardwares, a particular emphasis was placed on flexibility, expandability and modularity so that they may be expanded or modified easily.

ACKNOWLEDGMENT

The KORNET project has been entirely a team effort. The project which requires vast effort and states of the art technology would not have been successful without the dedication of our Ph.D. students of the Communication Research Laboratory at the KAIST. Particular thanks go to Dr. H. J. Cho who made a significant contribution in the early stage of the project.

REFERENCES

1. C. J. Weinstein et al., "Experience with speech communication in packet networks," IEEE J. Select. Area Commun., vol. SAC-1, pp. 963-980, Dec. 1983.
2. CCITT Recommendations, X.1-X.29, Yellow Book, vol. VIII, Geneva, 1980 and 1984.
3. 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.
4. L. Kleinrock et al., "Flow control: A comparative survey," IEEE Trans Commun., vol. COM-28, pp. 553-574, Apr. 1980.
5. M. Schwartz, "Performance analysis of SNA virtual route pacing control," IEEE Trans. Commun., vol. COM-30, pp. 172-184, Jan. 1982.
6. M. Schwartz et al., "Routing techniques in computer communication networks," IEEE Trans. Commun., vol. COM-28, pp. 539-552, Apr. 1980.
7. BTM Corp., "Packet switching system DPS25, operating and maintenance course," pp. 17-160, 1983.