National Library of Canada

Acquisitions and Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada

Direction des acquisitions et des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

A Token Ring Protocol for Integrated Services

by

Jian Song, M.Eng.

A thesis submitted to
the School of Graduate Studies and Research
in partial fulfillment of the requirement for the degree of

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

June 1995

ISBN 0-612-07808-6

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Abstract

In this thesis we present a new token-ring protocol, called the rotation counter protocol (RCP), for integrated services. The main feature of this protocol is that the token contains a special field called the *rotation counter*. This counter can be examined by every station and is used to count down the number of packets transmitted in one cycle. In each cycle, voice packets are collected first, and data packets could be transmitted only if the counter value is non-zero. In this way, voice packets are given a higher priority to access the network so that their real-time requirement can be satisfied. Two variations of the protocol, RCP-I and RCP-II, are presented. They have different delay performances. In RCP-I, the mean voice packet access delay is dependent on the offered data load, while in RCP-II, such delay can be made almost independent with a small delay jitters. The almost constant mean delay and small delay jitters can thus better serve synchronous traffic. Algorithms for the two protocols are given. They are easy to implement. The RCP performances are evaluated under various network conditions to demonstrate their desirable features. The comparison among the RCP-I, RCP-II, and FDDI is executed. The procedures on fault recovery and how to choose parameters are discussed as well.

Acknowledgments

Very special thanks are owing to my dedicated supervisor, Dr. Oliver W.W. Yang, whose expert advice and helpful support were vital in every stage of the work and preparation of this thesis, from the preliminary literature survey to the final examination. I also wish to express my gratitude to my colleagues in our computer communication research group for their valued discussions and support.

Most of all, I thank my husband for his consistent understanding, encouragement and unending moral support without which this thesis may never have been finished.

This work is financially supported by an NSERC Operating Grant under contract #OGP0042878.

Table of Contents

List of Figures

Table of Acronyms

CIV = Counter Initial Value
CV = Coefficient of Variation of the voice packet interdeparture time
FC = Frame Control field in the token
FDDI = Fiber Distributed Data Interface
IBNL = Individual Buffer and the Newest Lost
IBOL = Individual Buffer and the Oldest Lost
LAN = Local Area Network
LLC = Logic Link Control
MAC = Media Access Control
MAN = Metropolitan Area Network
RAT = Release After Transmission
RC = Rotation Counter field in the token
RCP = Rotation Counter Protocol
SBNL = Shared Buffer and the Newest Lost
THT = Token Holding Timer
TRT = Token Rotation Timer
TTRT = Target Token Rotation Time

Table of Symbols

L_{cycle}	=	cycle length in time units
L_d	=	data region in time units
L_v	=	voice region in time units
$\overline{L}_{message}$	=	mean data message length in number of packets
N	=	number of stations on the network
$N_p = 3$	=	threshold value when using limited service discipline
P_{trans}	=	one packet transmission time which is normalized to 1 time units
pv	=	voice packet length (including header) in bits
$\overline{R}_{message}$	=	data message arrival rate in messages per time unit per station
$T_{latency}$	=	ring latency of the ring in time units
$T_{interval}$	=	voice packet generation time in time units
$\overline{T}_{call}$	=	mean voice call holding time in time units
$\overline{T}_{idle}$	=	mean idle state duration in time units
$\overline{T}_{silence}$	=	mean silence duration in time units
$\overline{T}_{talkspurt}$	=	mean talkspurt duration in time units

Chapter 1

Introduction

1.1. Background

The principal technology ingredients that determine the nature of a Local Area Network (LAN) or Metropolitan Area Network (MAN) are [STAL93]:

- Topology
- Transmission medium
- Medium access control technique

Together, they mainly determine the type of data that may be transmitted, the speed and efficiency of communications, and even the kinds of applications that a network may support.

Four simple topologies are commonly used to construct LANs and MANs: bus, tree, ring, and star. The factors in choosing a transmission medium and a topology are not always independent of each other. By far the biggest advantage of a ring is that it so readily accommodates the use of optical fiber [ROSS87], because the ring consists of a series of point-to-point links, and the technology for point-to-point fiber transmission is well understood and widely available. The optical fiber ring is well suited for providing high data rates over long distances, better exploiting the potential of optical fiber. Due to the high data rates attainable with optical fiber, the fiber ring is a natural choice for a very high-speed LAN or for a MAN. High bandwidth and low loss over relatively long distances are major features of optical fiber.

High speed LANs and MANs have proven to be an attractive approach for transmission of data, voice, video, facsimile, graphics and other visual information. Integration of voice and data is a common form of integrated services. It may be considered as the first step towards a completely integrated office environment. The integration of digital voice with data in a single network offers potential cost savings through the sharing of switching and transmission resources. Significant channel capacity savings for digital voice transmission can be easily achieved on an integrated network by transmitting packets only when callers are actually talking. Such integrated networks can enhance the service flexibility for users who require access both voice and data communication.

Over the years, a number of different algorithms have been proposed for controlling access to the ring networks. The three common access techniques are [BUX89]:

(1) Slotted rings [WILK79, HOPP88]: several fixed-length data slots continuously circulate around the ring. Any station can place a data packet in one of the empty slots, along with the appropriate address information;

(2) Register insertion rings [HILA92, BUX83]: the conflict between data ready to be transmitted by a station and the data stream already flowing on the ring is resolved by dynamically inserting sufficient buffer space into the ring; and

(3) Token rings [IEEE85a, ANSI88, ROSS89]: access to the transmission channel is controlled by passing a special signal, the permission token, around the ring.

In these three ring-control schemes, all stations are peers and they autonomously determine when to transmit based on the state of the ring. This can be contrasted with other control schemes in which a single master station is responsible for controlling access to the ring or loop. An example of this is the FDDI-II where the Cycle Master station [STAL93] (or called Monitor Station [WILL92]) is sending a number of fixed size cycle structures on the ring. This cycle structure is subdivided into 16 Wide Band Channels which may be allocated to packet or isochronous traffic.

Token ring is a typical representative of ring networks and has a wide variation in Medium Access Control (MAC) protocols. The general approach is to seek an efficient MAC protocol that will optimize performance measures such as throughput and access delay. The well-known disciplines are:

(a) Exhaustive service [FERG85, SARK89]: the server continues to serve each queue

until it is emptied. Messages arriving at the queue in service are also served in the current service period.

(b) Gated service [HASH72]: the server serves only those messages that are queued at the polling instant. Those that arrive at a queue during the service to that queue are set aside to be served in the next round of polling.

(c) Limited service [FUHR88]: each queue is served until either the queue is emptied or a specified number of messages are served, whichever occurs first.

(d) Timed-token service [GROW82, ULM82, SPIE91]: the length of time the token may be held for transmitting frames of a given class depends on the time between successive arrivals of the token at the transmitting station.

In disciplines (a) to (c) above, the decision of a station to transmit a packet on the ring network is independent of the queueing status (e.g., the number of packets) at other stations. Thus the performance of a station, such as the access time for real-time voice, may be adversely affected by traffic from other stations. Performance comparison among these disciplines has been carried out [TAKA90]. In contrast, service discipline (d) correlates the performance of each station through a common parameter, the TTRT (Target Token Rotation Time), so that each station can be guaranteed to achieve a maximum access delay [SEVC87].

Much work has already been carried out to investigate service integration on token ring networks. For example, Wong and Gopal [WONG84] developed a token ring protocol based on the IEEE 802.5 standard for the packetized voice transmission. A supervisor station is needed to be a central controller to grant/deny voice call requests and set-up/tear-down voice calls. The supervisor station periodically issues a priority token to switch voice and data transmission. Wong and Yum [WONG89] showed a token-controlled ring protocol in which a *DS* field (data slot count) in the token is used to track the number of available slots for data in each frame and to ensure that the voice slots are not delayed by more than DS_0 slots. Choi and Krishna [CHOI90] presented an adaptive algorithm to ensure differential service in a token-ring network. The t.o.h. (token-owning-host), the node which currently holds the token, decides when to surrender the token by calculating priority indexes. Mark and Lee [MARK90] give a token-ring MAC protocol on a dual-ring. The access to the main ring is controlled by a set of four tokens which are circulating around the scheduling ring. The different tokens are used for different services which include voice, video, and data. Yang, et al., [YANG92] introduced a token ring protocol

by using two tokens, voice token and data token, to dynamically allocate bandwidth to data stations according to voice traffic load. Chen and Yu [CHEN93] proposed a token ring protocol which has modified the token format proposed in the IEEE 802.5 standard by appending a global counter (the dynamic counter), and a control bit (the recounting bit) to the free token to control the maximum number of voice calls on the ring. It also uses some counters in their protocol: a *dynamic counter*, appended to the free token, is used to control the number of voice call connections on the ring; two local counters which are kept at each network station, are used to control the data traffic flow (one of them is called a rotation counter!). The meaning and control method, however, are different from those used here.

FDDI [ANSI88, BUX89, ROSS89] applies the optical technology in the LAN environment. It can be viewed as either a high-speed LAN standard or as a MAN standard. FDDI employs a timed token protocol, in which the length of time the token may be held for transmitting frames of a given class (i.e., the token-holding time) depends on the time between successive arrivals of the token at the transmitting station (i.e., the token-rotation time). The FDDI protocol uses a number of timers and variables at each station to determine these times. This will be discussed later in Chapter 3.

1.2. Research Motivation

From discussion in the previous section, we know the following features are desirable when having voice and data (and other services) integration on token ring networks:

(1) The design of an integrated services protocol should satisfy the following requirements.

As we know, the voice traffic has two basic requirements: a small end-to-end delay and a synchronous output at the destination which could be expressed by zero delay jitters in the ideal condition. All integrated services protocols consider the first requirement as basic design criteria, but few of them discuss the second one (e.g., [WONG84]). Therefore, it would be useful to find a protocol that can satisfy both requirements which, in turn, will certainly improve the transmission performance of voice traffic.

(2) The performance of each station should be correlated through a common parameter, so that voice packets from each station can be guaranteed to achieve a maximum access

delay. This actually, in turn, satisfies one of the two basic voice traffic requirements.

All existing protocols have different approaches. For example, [WONG84] uses a central controlled supervisor station; in [CHOI90], the t.o.h. decides when to surrender the token by using an adaptive algorithm; [MARK90] and [YANG92] employ several tokens; and [CHEN93] modified the token format by appending a counter and a control bit to the free token. FDDI uses TTRT and several timers. We are interested in a different approach in this thesis.

(3) The transmitting station should pass the token immediately after the end of transmission to achieve high efficiency and easy adaptation to the optical fiber technology, such as FDDI. [WONG84] and [CHEN93] describe the instances based on the IEEE 802.5 protocol which requires the transmitting station to delay issuing a new token until the header of the transmitted frame has returned. .

(4) From the point of view of providing fault-tolerance, there should be no central controller in the network so that in the event of station failure, the system will not go down.

(5) There should be easy implementation.

1.3. Research Approach

On the basis of our interest in other variations of token ring protocols which can achieve the desirable features above, the Rotation Counter Protocol (RCP) is designed. It is hoped that this protocol has comparable or even better performance than the existing protocol. The results of our investigation, which have been presented in part in [SONG93a] and [SONG93b], will be presented completely in this thesis.

Unlike the FDDI timers that count down on time, our RCP introduces a special field called *rotation counter* which is used to count down on the number of packets transmitted in one cycle. Whereas each FDDI station keeps timers, each RCP station shares this global counter that is maintained in the rotating token. In this way, the performance at each station can be correlated through the counter value as discussed later. This counter can be examined by every station. In each cycle, voice packets are collected first, and data packets could be transmitted only if the counter value is non-zero. In this way, voice packets are given a higher priority to access the network so that their real-time requirement can be satisfied. Two variations of the protocol, RCP-I and RCP-II, are presented. They are

designed to have different delay performances. In RCP-I, the mean voice packet access delay is dependent on the offered data load, while in RCP-II, such delay can be made almost independent with a small delay jitters. The almost constant mean delay and small delay jitters can thus better serve synchronous traffic. Algorithms for the two protocols are given, and their performances are evaluated to demonstrate their desirable features. The algorithms are easy to implement.

It is of utmost importance that the newly designed protocol be capable of meeting the performance requirements. Performance evaluation has been shown to be a useful tool which adds a quantitative factor to the design process. It can also help us improve our design by uncovering any deficiency. In order to carry out our performance evaluation, we will construct a queueing model for our network operation, from which queueing analysis can be carried out. As queueing models becomes more detailed, the analytical complexity also increases. To investigate the properties and behaviors of the RCP, modeling and simulation is the most suitable approach [GARZ90]. Simulation also overcomes other limitations of analytic techniques; namely, peak loads and transient behavior can be investigated. Also by varying design parameters and buffering strategies, sources of performance fluctuations can be identified.

The importance of modeling and simulation is in the study of the following systems:

- impossibility of dealing directly with the system (e.g., the system may not yet exist).
- cost of studying the system directly, which may be too high.
- impracticality of dealing directly with the system (e.g., time-consuming).

Our RCP is a proposed protocol, so it belongs to the first case. We will use the modeling and simulation method to evaluate the network performance under this protocol.

Simulation is (loosely) considered to be the realization of a model, used to represent a system, in a form suitable for deriving information about the system's characteristics and properties of interest. Thus, one must first have a model of the system before a simulation can be developed. The system model itself can range from a basic conceptual model to a precise set of mathematical relationships describing the system's behavior. Based on the chosen representation, the associated model can be deterministic or stochastic in nature. Within each of these subdivisions, models can be further categorized in terms of the way in which their state changes as a function of time. The associated simulation can therefore be

classified into one of three types:

- continuous time: the simulation of a system of linear or non-linear differential equations.
- discrete time: the simulation of a system that is described by a set of difference equations.
- discrete event: arises from systems that can be modeled as a sequence of countable events, where it can be assumed that nothing of interest takes place between those events.

The third approach, discrete event simulation, deals with the mathematical modeling and simulation of systems subject to demands whose occurrence and lengths can, in general, be specified only probabilistically. Queueing and communications networks, telephone systems and computer systems are examples of such systems. All these problems share some common features, namely,

- probabilistic description of processes
- service time requirements
- waiting time for service (queue)

Furthermore, the performance of each can be measured using the same performance indices, such as response time, queue length, utilization, etc.

The proposed RCP is designed for token-ring network. It has all those features mentioned above, so we will use the discrete event simulation method to evaluate its performance.

In this study, we use QNAP2 (Queueing Network Analysis Package Two) as a simulation tool. QNAP2 is a highly structured, special-purpose simulation language oriented toward queueing networks; it is not a procedural language, such as FORTRAN or SIMSCRIPT. QNAP2 can be defined as a system for describing, handling, and solving *queueing network models*. The language of QNAP2 allows the user to describe the following items:

- the network configuration.
- the processing done by each station.

- the network resolution control.

Simulation is based on the constructed queueing model, and proper simulation times were selected to obtain convergent results and 95% confidence intervals within ±5% of mean values for each set of system parameters. Measures used in our performance evaluations include:

(1) Voice packet delay and delay jitters which measure the two basic voice traffic requirements respectively.

(2) Voice packet loss probability when the buffer is finite.

(3) Data message delay which is a useful quantity although not of primary importance.

(4) Network throughput used to identify desirable operation region.

Measures above are also critical to choose parameters for design.

Finally, our evaluation results will be used to suggest a design procedure for such a network.

1.4. Thesis Organization and Contributions

This thesis is organized as follows:

Chapter 2, System Description and Operations, outlines the token ring network, station functional structures and the information packetization procedures.

Chapter 3, Rotation Counter Protocols, explains two variations of the RCP, RCP-I and RCP-II, so that the transmission cycle structure is determined; gives the token format and packet format under the RCP; demonstrates the detailed functions of the user station and the supervisor station under the two variations of the RCP.

Chapter 4, Model Description, describes the voice and data arrival processes, the buffering schemes at each station, the system queueing model, and the QNAP2 model.

Chapter 5, Performance Evaluation, gives the performance measures and simulation results. Performance evaluation is carried out in several cases: For the first variation of RCP, RCP-I, we discuss its performance under different disciplines and under different

network situations. When the network is attached with a single station, we evaluate the RCP-II performance and compare the RCP-I and RCP-II with FDDI. After that, the backbone network, using RCP-II, is evaluated when voice call has infinite or finite size of the buffer pool. Different buffering schemes for the finite buffer are discussed. Various network situations such as the transmission cycle length, the network size, voice traffic load, and network load, are evaluated. An example when the network is working under the asymmetric condition is given.

Chapter 6, Design Parameters and Implementation, summarizes the simulation results and indicates how to choose some parameters. It also describes the fault recovery procedures.

Chapter 7, Conclusion, presents the concluding remarks.

The main contributions of this thesis lie in the following:

1. The proposal of a new MAC protocol, the Rotation Counter Protocol, which is designed for integrated services on a high-speed token-ring network.

2. The formulation and description of two RCP algorithms, the RCP-I and RCP-II, for different service integration purposes.

3. The modeling of the system to describe the voice and data traffics arrival procedures, the transmission control method, and the whole network operation.

4. The performance evaluation of RCP-I and RCP-II under different disciplines, diverse network parameters, changed traffic loads, and variable buffer conditions.

5. The comparison with the timed-token protocol which is employed in FDDI and has similar functions as our RCP. Except for the MAC protocol, this is done under exactly equivalent network conditions.

6. The design and comparison of various buffering schemes when the voice packets are waiting for the transmission and the buffer sizes are limited.

7. The demonstrations that the RCP can achieve some desirable features, such as load-independent voice packet delay and voice packet interdeparture time, low delay jitters, "floating supervisor" scheme, and it can be easily extended to adapt to the new requirements (e.g., the RCP-II is the extension of the RCP-I).

Chapter 2

System Description and Operations

As stated in the previous chapter, our RCP is designed for integrated services on a high-speed token ring network. Here we will outline the token ring network and its operation, station functional structures, and voice/data packetization procedures.

2.1. The Network and its Operation

In a general token-ring network, stations are assumed to be numbered sequentially from 1 to *N*. They are connected by unidirectional transmission links to form a closed path. Permission to transmit at a station is granted by a token, which is a control signal composed of a unique signaling sequence and being passed around the ring continually. There shall be only one token in the network at a particular time point so that no collision occurs. Only the station capturing (and holding) the token may hold the maximum allowed amount of time to transmit. While not holding the token, a station would merely repeat the incoming symbol stream onto the output link. Each receiver regenerates and repeats the data packets and the token at their original signal strength, so that networks are not limited by distance or speed as are bus-type networks. Only the addressed destination station copies the information as it passes, and only the station which has transmitted the information removes it from the ring.

In this study, optical fiber is used as a transmission medium to allow broadband communication, and a Release After Transmission (RAT) MAC protocol [WALR91] is used so that the token is released upon the completion of packet transmission at a station. When a station wants to transmit a frame, it follows the following steps:

1. Waits until it detects the token.

2. Captures the token.

3. Stops the token repeat process. (Since there is no token on the ring, this action prevents other stations from transmitting data onto the ring.)

4. Begins sending frames. (Frames can be sent until there is no more data to send, and/or the token holding rules require surrender of the token.)

5. Releases the token onto the ring for use by another station.

Note that the stripping function differs from IEEE 802.5. In 802.5, a transmitting station keeps the ring open until its complete packet has returned, and here the source address of the packet has to match the station's own address to trigger stripping.

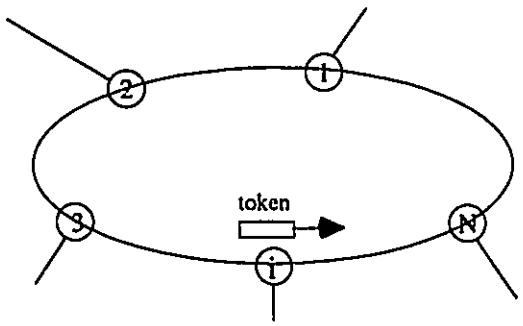


Fig. 2.1 Configuration of a Token Ring Network

2.2. Services on the network

Our protocol is designed for integrated services. Different types of traffic have different service requirements. In terms of traffic to the integrated LANs, they can be classified into [WONG89]:

(1) Short asynchronous data traffic: examples are interactive terminal inquiries, remote control messages, and sensing messages.

(2) Bulky asynchronous data traffic: these are the machine-to-machine traffic with large volume of data transfer such as file transfer, database updating, and high-volume printing.

(3) Synchronous stream traffic: these are generated in real-time person-to-person calls and hence cannot tolerate a large transmission delay. Voice is an example. Video is similar to voice except at a higher bandwidth.

For simplicity, we assume that each station on the network is active in both voice and data traffics.

Therefore, an integrated token ring network is shown in Fig. 2.2.

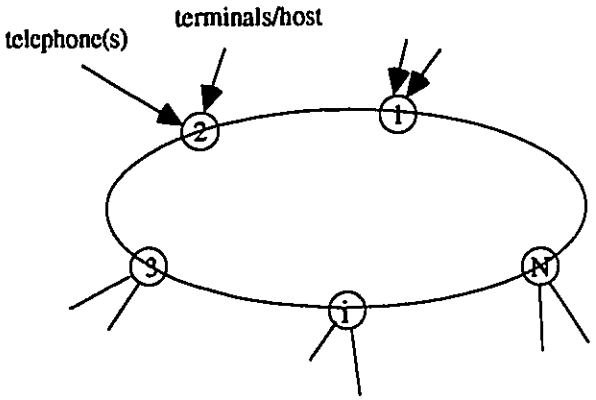


Fig. 2.2 An Integrated Token Ring Network

2.3. Packetization Procedures

On a token ring network, voice and data information is transmitted in fixed length packets. For information length greater than the designed quantum, e.g., voice talkspurt, packetization must be used. The packetization procedures are presented as follows.

Fig. 2.3 demonstrates the connection of voice users to the network [SUDA89, WONG89]. At each voice source, a continuous voice analog signal is digitized by a coder. The generated samples are accumulated in a packetizer. When the number of samples in the packetizer reaches the predetermined packet length, a header is attached and a voice packet is generated. The generated voice packet is then examined by a speech activity detector to see if it contains some minimal level of speech activity. Silent packets are discarded. Non-silent packets are stored in the buffer in the order of their generation, and await transmission. The packet generation cycle is independent of the packet transmission process. Therefore the queue size at the buffer may continue to grow while a packet is

waiting for transmission. In general, packets may be lost if the buffer size is finite.

The voice signal is transmitted in the form of packets where each packet consists of a number of voice samples within a packetization interval. Voice packet delays through the network are measured upon arrival at the destination user. The packets are disassembled into individual samples to reconstruct the speech. Through D/A decoder, the receiver obtains the analog signal.

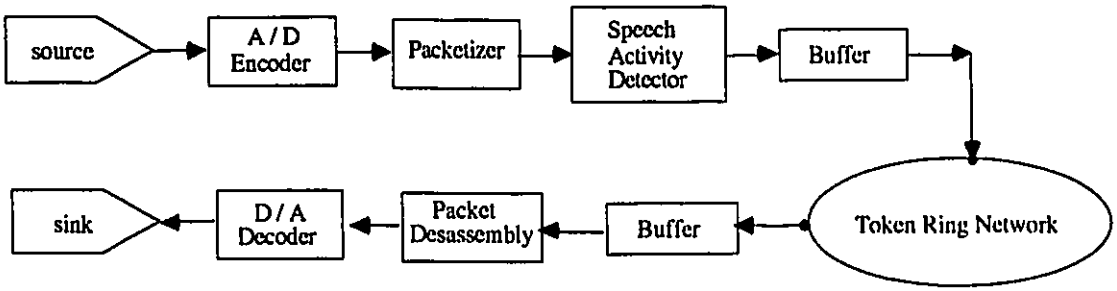


Fig. 2.3 A Voice Call on the Network

The processing of a data message is represented in Fig. 2.4 [BUX85]. A sending (source) station generates messages with given length and interarrival time distributions. Higher layers perform the function of "high-layer interface" on top of LLC (Logic Link Control), and the message segmentation/reassembly functions when the maximum packet length specified for LLC is exceeded.

User data to be sent are passed down to LLC, and a header is appended. This header contains information that is used to manage the protocol between the local LLC entity and the remote LLC entity. The combination of user data and LLC header is referred to as a data unit. It is then passed as a block of data down to the MAC entity. The MAC entity appends both a header and a trailer, to manage the MAC protocol. The MAC-level data unit is typically referred to as a frame. The frame is transmitted to, and received from, the ring network.

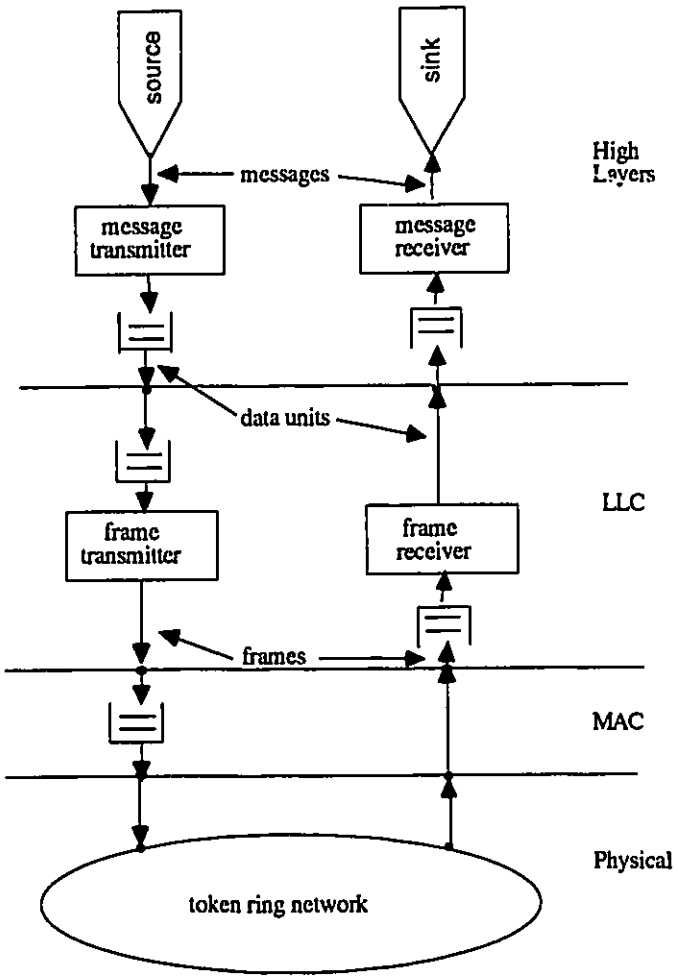


Fig. 2.4 A Data Message on the Network

Due to the design of our protocol, both voice and data packets are to take on the same constant length. Fixed length packets are our protocol requirement because the rotation counter is to count on the number of transmitted packets. Fixed length packets can also provide some advantages which are partially reflected in the ATM standard [DePR91]:

(1) Queue memory management: in the case of fixed length packets, the memory management system can assign memory blocks with always the same size, namely the size of the packets. This operation is rather simple, as is the management of the free memory list, which is only packet based.

(2) Queue memory size requirements: the memory requirements of a system based on fixed length packets depend on the load and the acceptable packet loss rate. The queue dimensioning has to be done in number of packets. In bytes this means that the larger

the packet, the larger the memory requirements are. In case of variable length packets, the queue dimensioning is much more complicated and depends on the mix of packet lengths.

2.4. Backbone Network

A token ring network can work as a simple network, for example, one phone and one terminal per station. In this case, we assume that no two voice calls arrive at a certain station simultaneously. Our RCP is designed for a high-speed token-ring network. It should be able to work as a backbone network for network interconnections. Each station could perform functions of a bridge or a concentrator, which means it may have multiple voice call arrivals and the data messages may come from different nodes which are connected to this station. Here we will not discuss the translating address and protocol between very dissimilar networks but traffics only. We assume that M voice calls arrive at this station and the data message arrivals for this station are the total of all data messages from all connected nodes. The functional block diagram of station *i* is shown in Fig. 2.5. Obviously, the simple network can be considered as a special case of backbone network with M=1.

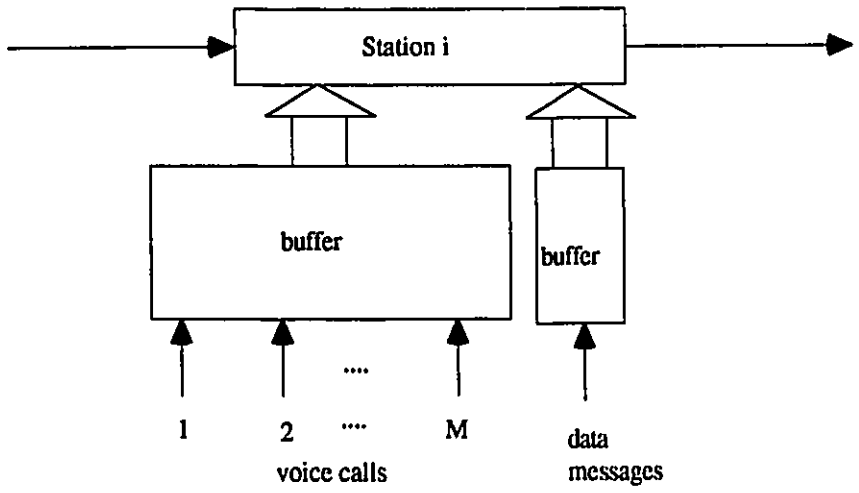


Fig. 2.5 A Functional Block Diagram of Station i

Overall, our protocol is designed for a token-ring network with N stations. It can

work as a simple network (M=1) or a backbone network (M>1). Each station has two kinds of traffics: voice and data. All the messages are packetized with the same fixed length. The transmission on the token ring is controlled by the token which is being passed around the ring continually.

Chapter 3

Rotation Counter Protocols

The proposed rotation counter protocol is designed for integrated services on a token ring network based on the standard FDDI medium access method described in [ANSI88] and [STAL93]. For simplicity, we shall use voice-and-data integration as an example for all discussions in the following.

The major difference between the RCP and other token-ring protocols is that a special field is included in the token. This special field contains a counter that would count down on the number of packets transmitted in a cycle. At the beginning of a cycle, the counter is initialized by a supervisor station with an agreed-on threshold called the Counter Initial Value (CIV).

After capturing a token, a station will decrement the counter by one for each packet it has transmitted. If a station ends its packets while the counter is still greater than one, it passes the token (containing the updated counter value) to the next station and the next station will transmit its packets by continuously using this counter. A station has to stop transmitting once the counter becomes zero and to pass on the token. The next (downstream) station that receives a zero counter will reinitialize the counter to CIV before its own transmission operation. Then a new transmission cycle starts[1].

In each transmission cycle, the voice packets from every station are first collected. If the counter is not zero after voice transmission, data packets, if present, are transmitted according to certain rules (such as those given in RCP-I and RCP-II which will be discussed later). As soon as the counter becomes zero, the transmission will be for voice

---

1 This is different from the definition found in other polling analyses. It is obvious that a cycle here may not go through every station if the network load is heavy.

packets again. In this way, voice packets are given a higher priority to access the network and the two successive token visiting time is bounded. Therefore, voice packet real-time requirement can be satisfied.

3.1. Token and Packet Formats

As discussed in the previous chapter, the token is a control signal whose format is shown in Fig. 3.1. The functions of each field are designated as follows:

SD = Starting Delimiter
FC = Frame Control
RC = Rotation Counter
ED = Ending Delimiter

Fig. 3.1 Token Format

(a) Preamble: A fixed pattern to synchronize the frame with each station's clock. The originator of the frame uses a field of 16 idle symbols (64 bits); subsequent repeating stations may change the length of the field consistent with clocking requirements.

(b) Starting Delimiter: A unique 8-bit pattern to indicate the beginning of a frame.

(c) Frame Control: A field with a bit format 'CFFASLRR' where

- C indicates whether this is a synchronous (C=0) or asynchronous frame (C=1);
- FF indicates whether this is an LLC (logical link control), MAC (medium access control), or a reserved frame. A token frame is indicated by FF=00. The A-bit indicates the size of the counter field: A=0 for a16-bit field and A=1 for a 48-bit field;
- S indicates whether the supervisor station changes (S=0) or not (S=1);
- L indicates whether voice packets from all the stations are collected in one cycle (L=0) or not (L=1);

- RR is reserved for future use.

(d) Rotation Counter: A count-down counter field which indicates the remaining number of packets allowed to transmit in this cycle.

(e) Ending Delimiter: A unique 8-bit pattern to signal the end of frame.

The actions taken by each station in response to (c) and (d) above will be further discussed further later in this chapter.

Any data packet formats suitable for high speed ring networks can be used in this protocol. However, for completeness purpose, we include the format of a data packet that is adapted from the FDDI standard. The format is shown in Fig. 3.2. The details of different fields can be found in the token format and [ANSI88]. The information field length is fixed, which is different from the FDDI standard.

SD = Start Delimiter
FC = Frame Control
DA = Destination Address
SA = Source Address
INFO = Information
FCS = Frame Check Sequence
ED = Ending Delimiter
FS = Frame Status

Fig. 3.2 Packet Format

3.2. Transmission Cycle Structure

As we mentioned before, on the token ring network, a station with packets to be transmitted must wait until seizing the token on the transmission medium. This implies that the time interval between two consecutive token arrivals will dominate the packet delay. For real-time consideration, the traveling time of the token must be controlled well. We define a *transmission cycle* as a period of time needed for the token to circulate around the ring and serve all the network stations, and we let the transmission cycle time duration be L_{cycle}. Fig. 3.3 gives a typical RCP transmission cycle at any point on the network.

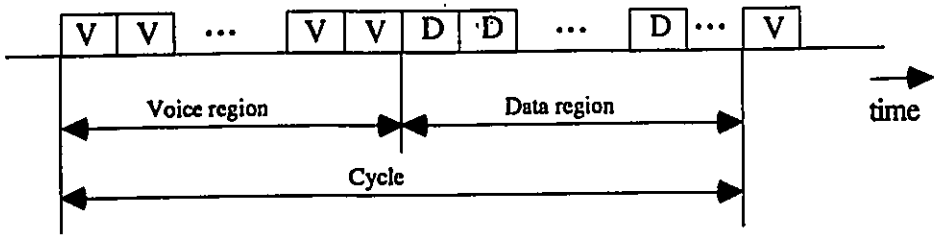


Fig. 3.3 A Typical Transmission Cycle Structure in RCP

A cycle is divided into two regions: the voice region and the data region, and the boundary between them is moveable. Each cycle begins with the voice region as the token makes one round collecting voice packets. Let the time duration of the voice region be L_v. Obviously, $L_v \geq 0$. Note that this duration depends on the number of voice packets generated during the previous cycles; it is otherwise independent of the offered data load.

The data region makes up the rest of the transmission cycle. Let the time duration of the data region be L_d. Then, obviously, $L_d = L_{cycle} - L_v$, and $0 \leq L_d \leq L_{cycle}$. The value of L_d determines what is to be collected after the voice region, and there are two cases:

(1) $L_d = 0$: the voice region occupies the full cycle length, i.e., $L_v = L_{cycle}$. It implies that the voice load is heavy at the present cycle, and there shall be no service for data in this cycle. The token will rotate to collect voice packets again.

(2) $L_d > 0$: there is room for data in the present cycle. So after collecting voice packets, the token will rotate to collect data packets.

Clearly, as long as the length of each transmission cycle L_{cycle} is well controlled, the real-time delivery of voice packets can be carried out easily. The value of L_{cycle} depends on several parameters: the value of CIV, the data load and the voice load.

3.3. Variations of RCP

If restricted to considering the real-time delivery of voice packets, we could only control the maximum length of the transmission cycle L_{cycle}. On the other hand, it is not true that the smaller the voice packet delay is, the better its performance will be. Based on two

basic voice traffic requirements, as long as the voice packet delay is kept within a certain range, a synchronous output of voice packets is desired. It may be preferable to keep the L_{cycle} constant. Therefore, we have two choices: either we have a variable but capped cycle length L_{cycle} which depends on the offered data load, or we have a constant cycle length L_{cycle} which does not vary with the offered data load. Based on these two choices, two variations can be formulated, the RCP-I and the RCP-II. We will discuss the design of these two variations.

3.3.1. RCP-I

In this protocol, L_{cycle} is varied with respect to the offered data load, and consists of the duration of one voice round and at most one data round. Its length is bounded by CIV when all packets have fixed transmission time. Note that the data round may not go through every station if the offered network data load is heavy. In this case, L_{cycle} will take on its maximum duration, and the data round is terminated prematurely. In general, a longer L_{cycle} gives a larger voice packet delay, and the voice packet access delay increases as the offered data load increases. However, the CIV has the effect of capping the maximum delay at a high data load.

3.3.2. RCP-II

In this variation, L_{cycle}, is fixed at the maximum that is determined by the CIV value. The cycle duration consists of one round for voice packets and as many rounds as there can be for data until the condition $L_d = L_{cycle} - L_v$ is satisfied. The idea is to have the fixed transmission cycle length be the same as the voice packet interarrival interval, so that two advantages might appear, the first is to keep the voice delay constant, and the second is to achieve hopefully a small delay jitters.

Note that if the offered data load is very heavy, the round for data would still terminate earlier when the counter counts down to zero. On the other hand, if there is no data at all, the number of data rounds can be infinite. To prevent this from happening, we take the propagation delay into account. The implementation detail will be discussed in section 3.4.2.2. First, we shall discuss the functions of the network stations.

3.4. Network Stations and Operations

All network stations can send and receive information. Each station performs the function discussed in chapter 2. A waiting packet of a particular class is considered to be an *eligible* packet when the token is within the region of the same traffic class.

On the network with N stations, one station is serving as an active supervisor station while all other active stations take on the functions of the stand-by supervisor and are called user stations. Stations are classified according to their functions and the algorithm employed while holding the token. They are described in the following sections.

3.4.1. User Station

A station wishing to transmit a packet must wait until the token has arrived. After detecting the preamble and SD (starting delimiter) fields, the contents of the FC (frame control) field and the RC (rotation counter) field of the frame are examined according to the algorithm in Fig. 3.4, where C and S are the control bits in the FC field and the counter is the value in the RC field. The actions taken by the user station are summarized in the following:

(1) C=0, counter>0: This indicates that voice packets are eligible for transmission. The present station is allowed to transmit its waiting voice packets, if any.

(2) C=0, counter=0: This indicates that voice packets are eligible packets, but not all stations have completed their voice transmissions during the cycle. The present station sets L=1, and transmits its waiting voice packets, if any[2].

(3) C=1, counter>0: This indicates that data packets are eligible packets. The present station transmits its waiting data packets, if any.

(4) C=1, counter=0: This indicates that a normal cycle(one that has both voice and data regions) has ended. The present station sets C=0 and counter=CIV, and assumes the supervisor function by setting S=1, and then transmit its voice packet, if any.

The present user station decrements the counter in the RC field by a number equal to

---

2 Note that this step may not happen, e.g., if CIV>N and each station is allowed one voice packet transmission.

the number of packets it has transmitted, and then passes the token immediately to the downstream station.

From the description in (4), it is obvious that each user station can be a supervisor, so we call it a "floating supervisor". But there is one and only one supervisor at one time. The advantage of this distributed operation is to make the network more reliable.

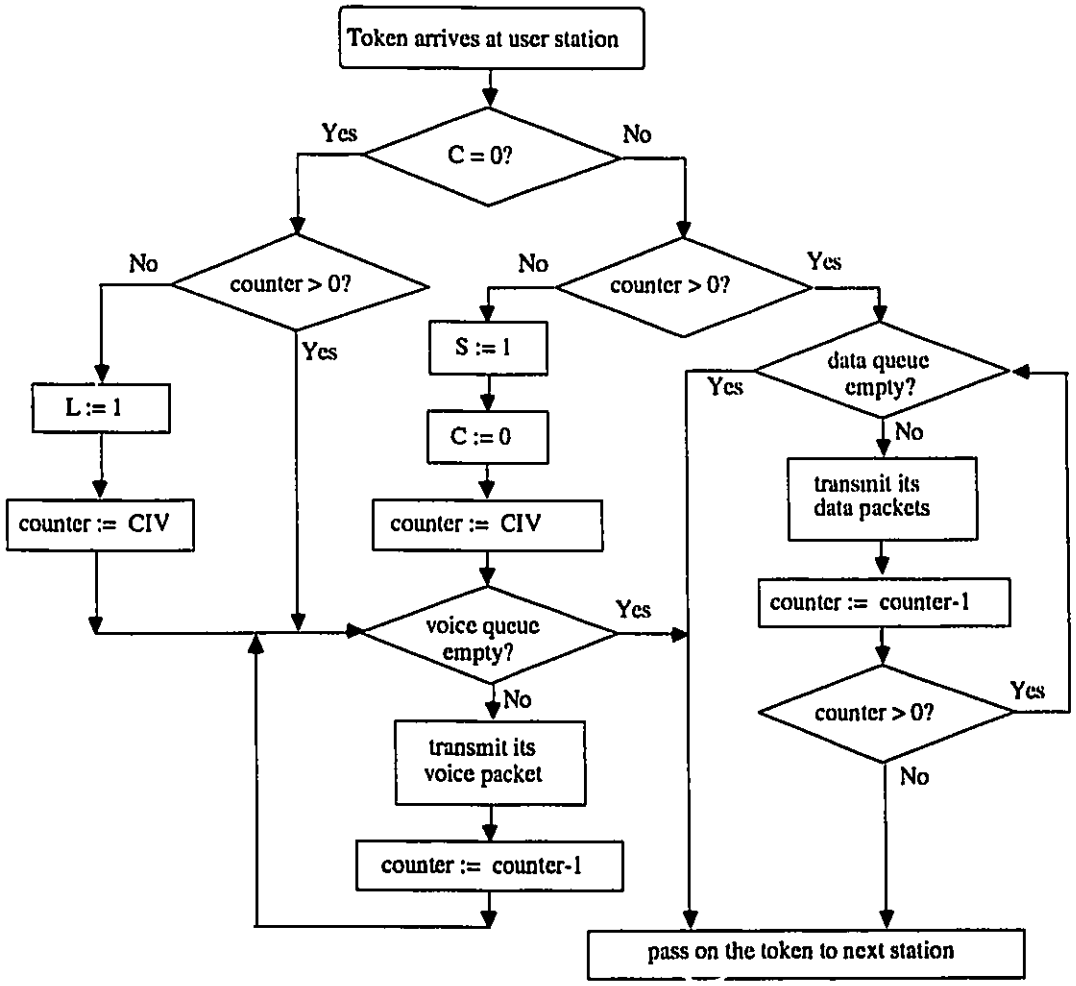


Fig. 3.4 User Station

One point requiring discussion is how many voice packets or data packets should be transmitted during the token's visiting. This depends on many conditions, such as the relationship between the voice packet generation time and the length of transmission cycles and whether or not the network is a single network or a backbone network. For example, if the voice packet generation time is equal to the length of the transmission cycle, one voice

packet per voice call is generated per transmission cycle, and there will be M voice packets per station. We will find out the solution from the performance evaluation.

Another item to be noticed is that the functions of the user station are the same for both RCP-I and RCP-II.

3.4.2. Supervisor Station

In addition to the packet transmission function of an ordinary user station, the supervisor station is also responsible for managing transmission cycles on the ring. The supervisor station also performs the monitoring function. It alone is responsible for seeing that the ring operates correctly. It detects faulty or incomplete tokens, detects lost token or frames, clears the ring when faults occur, and generates a new token, and monitors more than one active supervisor on the ring. These will be discussed in Chapter 6.

As mentioned earlier, two RCP variations can be achieved by manipulating the FC and RC fields in the token in cooperation with the user stations. Although the function of the user station is the same for different variations of RCP, the supervisor station functions are distinct. Therefore, a flow chart is provided for each of the RCP-I and the RCP-II. The supervisor's action in each protocol is summarized in the following.

3.4.2.1. Supervisor Station under RCP-I

As stated previously, the C, S, and L bits are the control bits in the FC (frame control) field of the token and the counter value is in the RC (rotation counter) field of the token. If a user station receives a token with C=1 and counter=0, it becomes a new supervisor station by setting S=1. However, if a user station receives a token with C=0 and counter=0, it sets L bit to 1 without changing the supervisor station. The supervisor station in RCP-I examines the receiving token and takes the following actions according to the algorithm in Fig. 3.5.

(1) S=1: This indicates that the supervisor station has already changed hands. The present station thus relinquishes its supervisor function and resets S=0.

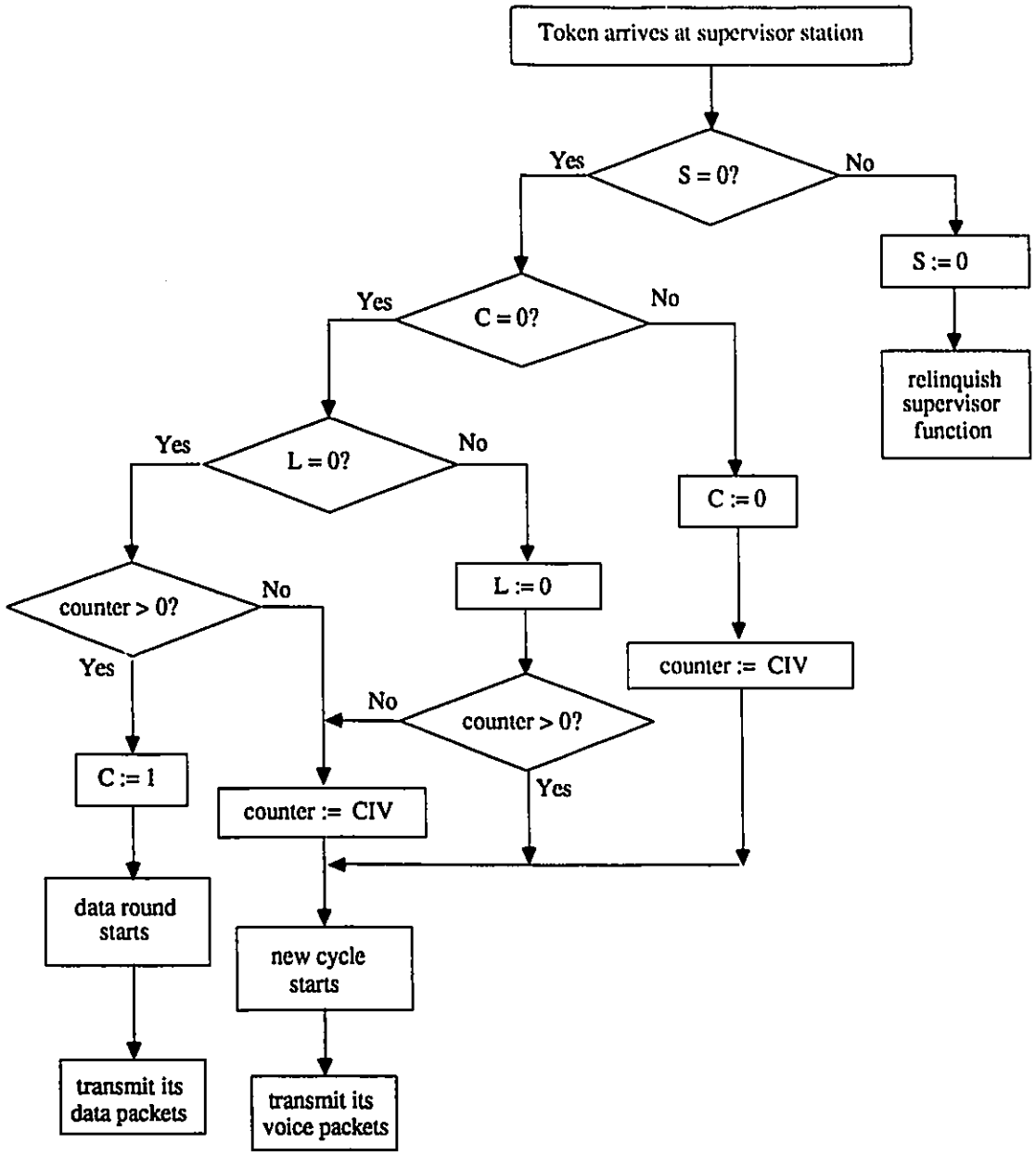


Fig. 3.5 Supervisor Station of RCP-I

(2) S=0, C=0, L=0, and counter>0: This indicates that the token has just finished the round for voice, and there is still room left for data. The supervisor station sets C=1 and begins to transmit its own data packets.

(3) S=0, C=0, L=0, and counter=0: This indicates that the token has finished a round for

voice packets, but there is no room left for data. The supervisor station should start a new cycle by setting the counter=CIV. It then can transmit its own voice packets.

(4) S=0, C=0, L=1, and counter>0: This indicates that the last cycle did not complete one round for all the voice transmission. The supervisor station sets L=0, and begins the transmission of its own voice packets.

(5) S=0, C=0, L=1, and counter=0: This indicates that the last cycle did not complete all the voice transmission, and a new cycle should start. The supervisor station sets L=0 and counter=CIV, and begins the transmission of its own voice packets.

(6) S=0, C=1: This indicates that the token has finished the round for data, and a new cycle should start. The supervisor station sets C=0 and counter=CIV, and begins the transmission of its own voice packet.

3.4.2.2. Supervisor Station under RCP-II

Recall that RCP-II modifies on RCP-I by keeping a constant cycle length. This can be achieved easily by modifying the RCP-I algorithm when the token is received with S=0, C=1. In this case, the supervisor station will check the counter value first. If counter=0, the supervisor will set C=0 and counter=CIV. A new cycle then follows. Otherwise, the supervisor will keep C=1 and transmit its own waiting data packets, if any, and the token will rotate for another round of data.

In order to prevent the token circulating indefinitely for data round in the absence of data packets, the supervisor station will limit the number of rotations so that a new cycle will start at the end of the duration L_{cycle}. To do this, the supervisor station needs to maintain a counter R at its location to count on how many times the token has circulated around the ring. Let $T_{latency}$ be the ring latency in time, and P_{trans} be the packet transmission time. Then the time that the token circulates $P_{trans}/T_{latency}$ times around the ring is equivalent to one packet transmission time. Therefore, if the token has circulated around the ring for $P_{trans}/T_{latency}$ times, the RC counter should be decremented by one, and the counter R should minus $P_{trans}/T_{latency}$. This can be concluded by the following simple algorithm:

- the R is updated according to

$$R_2 = R_1 + 1 - \frac{P_{trans}}{T_{latency}} \times \left\lfloor \frac{T_{latency} \times R_1}{P_{trans}} \right\rfloor .$$

- the RC value is updated according to

$$counter_2 = counter_1 - \left\lfloor \frac{T_{latency} \times R_1}{P_{trans}} \right\rfloor .$$

In the above, $\lfloor\ \rfloor$ is the floor operator, and subscript 2 indicates the new value while subscript 1 indicates the old value.

The flow-chart of RCP-II is given in Fig. 3.6.

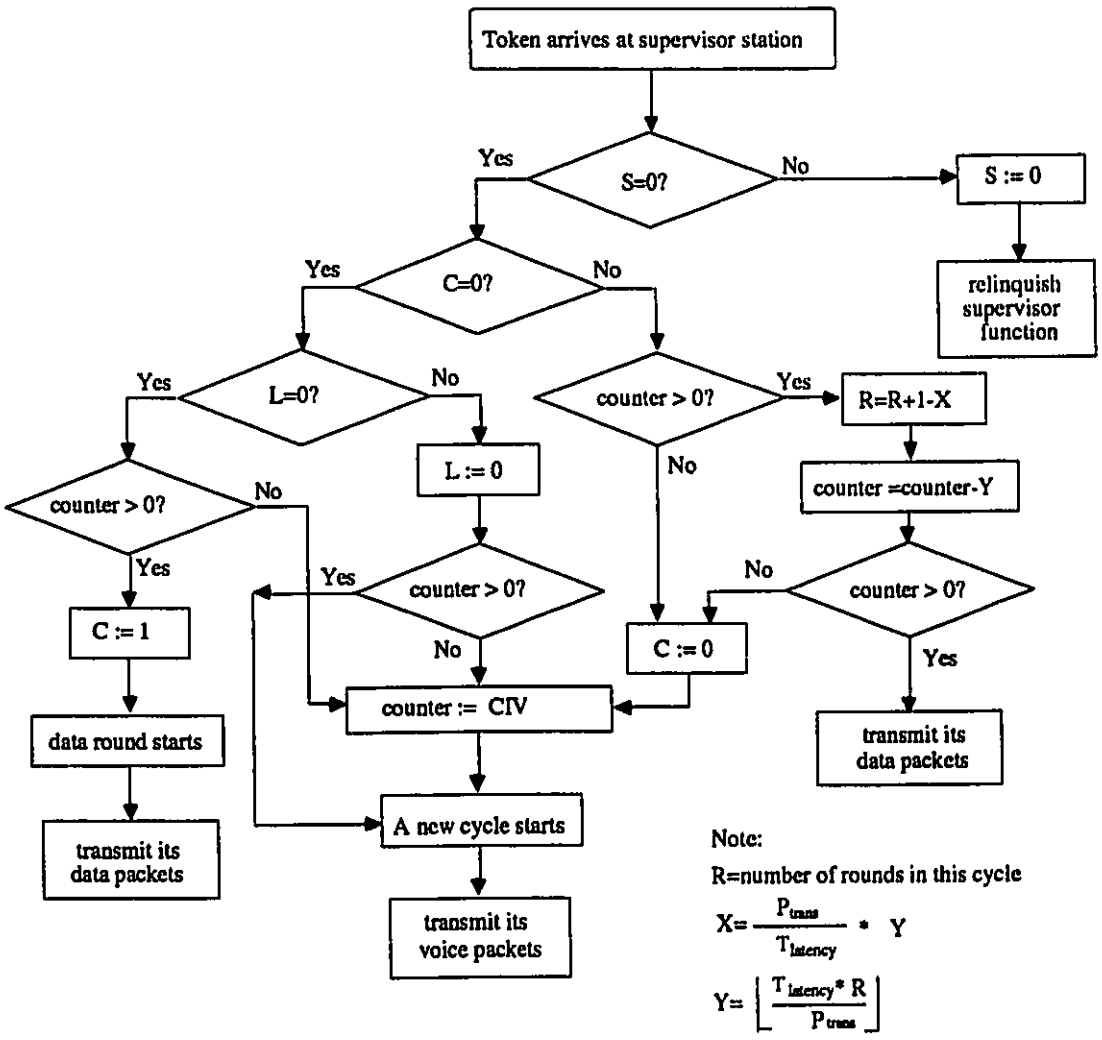


Fig. 3.6 Supervisor Station of RCP-II

Altogether, two versions of the RCP are designed: RCP-I and RCP-II. RCP-I can have a variable but capped cycle length which depends on the data traffic load. RCP-II will give a constant cycle length which is independent of the data traffic load. The cycle length will determine the voice packet delay.

3.5. Comparison of RCP with FDDI

3.5.1. FDDI Standard

In FDDI, two classes of traffic are defined: synchronous and asynchronous. The different classes are managed by the timed-token protocol. This protocol guarantees that stations will gain access to the ring within a time period that is negotiated between the stations each time a new station joins the ring.

FDDI employs a timed-token protocol in which the length of time the token may be held for transmitting frames of a given class (i.e., the token holding time) depends on the time between successive arrivals of the token at the transmitting station (i.e., the token-rotation time). The FDDI protocol uses a number of timers and variables at each station to determine these times. The important timers are TRT (Token-Rotation Timer), used to measure the time between successive arrivals of the token at that station; and THT (Token-Holding Timer), used to control the amount of time the token is held for transmitting asynchronous frames.

A Target Token Rotation Time (TTRT), negotiated during the ring initialization process, can be adjusted to satisfy different bandwidth and delay requirements. The TTRT is important for two reasons: (1) it sets the limit (bounded delay) on the time between successive token opportunities for any station to $2 \times TTRT$, and (2) it sets the upper bound on the available synchronous bandwidth to $\frac{TTRT}{TTRT + t_{latency}} \times 100$ Mbps. The availability of asynchronous transmission time depends on the ring load conditions and is also affected by the TTRT.

When a station receives the token, its actions will depend on whether the token is early or late. If the token is early, the station saves the remaining time from TRT in THT, resets TRT, and enables TRT:

THT ← *TRT*
TRT ← *TTRT*
enable TRT

Then it transmits synchronous frames. After transmitting or if there were no synchronous frames to transmit, THT is enabled. The station may transmit asynchronous frames only as long as THT > 0.

If a station receives a token and the token is late, the late counter is set to zero, and TRT continues to run. The station can then transmit synchronous frames, and may not transmit any asynchronous frames.

3.5.2. Comparison

Both the FDDI standard and our RCP employ a token protocol which allows the transmitting station to pass the token immediately after the end of frame transmission (i.e., they are all using RAT MAC protocol).

Both of them can make use of silence periods existing in the voice traffic to transmit data packets without any extra demands, because they do not allocate a fixed slot for a particular pair of voice calls.

Both of them are designed to provide different classes of service that enable the network to simultaneously support traffic with different transmission requirements. The RCP-I are very close to the FDDI: they only limit the upper bound of transmission delay and do not consider the CV requirement of voice traffic. The RCP-II consider both delay and CV requirements. Although FDDI-II, an upward-compatible extension to FDDI, can solve this problem by adding the circuit-switch ability to support real-time voice and video traffic, the RCP provides a packet-switched operation for all traffic types.

The methods for controlling data transmission are different in several aspects between two token-passing protocols. The timed-token protocol in the FDDI uses two timers to limit the amount of data transmission from each station: THT and TRT. Data service is allowed only when the time duration since a token was last received has not exceeded a threshold called the TTRT, that is, THT>0. The cycle length (one repeated period) is bounded by TTRT. On the other hand, our RCP allows packet transmission only when the counter in the RC field of the token is non-zero. The cycle length is controlled by CIV.

The transmission orders are also different between the two protocols. With reference to Fig. 3.7(a), data packet transmission (D) in FDDI always follows voice packet transmission (V) from the same station (as long as THT>0). The transmission order goes from station 1 to *N*. On the other hand, our RCP will collect voice packets (V) from every station before data (D) is allowed in (Fig. 5.6(b)). After finishing voice transmission for all stations, the data transmission round starts at the station where the voice round has ended.

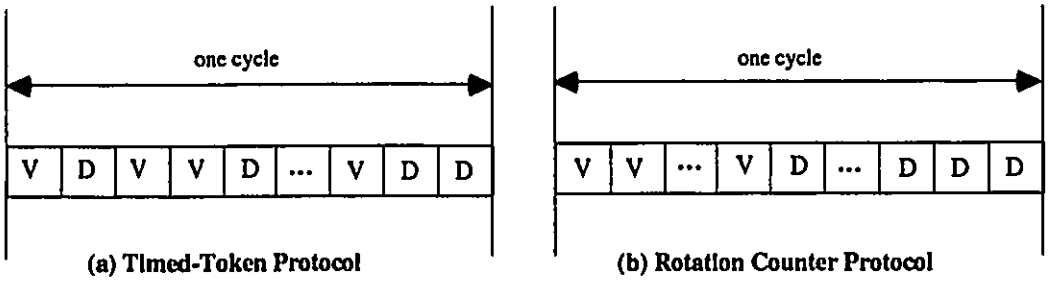


Fig. 3.7 Comparison of Transmission Order in FDDI and RCP

The transmission cycle lengths of the FDDI and RCP-I have the maximum value limitation only. They could be very short when traffic loads are very light. But that of the RCP-II is almost fixed which is independent of traffic loads.

The above discussion describes the RCP protocol and network operation. It also gives the similarities and differences between the RCP and the timed-token protocol on FDDI. The following chapters will model this system and evaluate both RCP-I and RCP-II performances. We will also compare the performances of RCP with FDDI.

Chapter 4

Model Description

As stated in Chapter 1, the performance evaluation will be accomplished by the modeling and simulation method. This method to evaluate performance has three steps: performance profile; simulation model (computer code form); and simulation evaluation results. The performance profile consists of three major models:

(1) workload model: to specify the characteristics of the resource demands on various equipments in the system.

(2) configuration or system structure model: to specify the hardware characteristics of the system.

(3) scheduling model: to specify the scheduling algorithms whereby resources are allocated.

We will describe performance profile in this chapter and give the simulation results in next chapter.

4.1. Voice and Data Traffic Model

The characteristics of voice traffic are quite different from that of data traffic [KARV86, WONG84]. The detailed description and their traffic models are as follows.

Voice traffic is a continuous bit rate service and it exhibits ON-OFF characteristics. With reference to Fig. 4.1, each voice source alternates between an on-hook state and an off-hook state. A voice call is assumed to be in progress in the off-hook state; it is idle otherwise. In a typical conversation (in the off-hook state), voice traffic capacity in each direction is idle for more than 60-65 percent of the time [BRAD59].

As discussed in section 2.2.3, packetization procedures, at each voice source, a continuous voice analog signal is digitized by a coder. The generated samples are accumulated in a packetizer. The generated voice packet is then examined by a speech activity detector to discard silent packets. Non-silent packets are stored in the buffer and waiting await transmission.

If a constant sample rate is employed and the packet length is predetermined, a voice call can, therefore, be modeled as alternating between talkspurts and silences with packets generated at a fixed interval of $T_{interval}$ during talkspurts and no packets during the silences shown in Fig. 4.1. This is sometimes referred to as the ON-OFF model, e.g., [OKUD92]. All voice stations are assumed to use the same voice digitization rate and therefore have the same packet generation time constant $T_{interval}$. Let p_v be the length in bits of each voice packet generated. Voice call arrivals are assumed to be Poisson distributed. Both voice calls and idle periods are assumed to be exponentially distributed with a mean of $\overline{T}_{call}$ and $\overline{T}_{idle}$ time units respectively. Similarly, the talkspurts and silence periods are also assumed to be exponentially distributed with a mean length of $\overline{T}_{talkspurt}$ time units and $\overline{T}_{silence}$ time units respectively.

For an ON-OFF model, the mean arrival rate of voice packets is given by:

$$\frac{\overline{T}_{talkspurt}}{T_{interval} \times (\overline{T}_{silence} + \overline{T}_{talkspurt})} \tag{1}$$

The mean arrival rate of voice packets per voice source, λ_{voice} is

$$\lambda_{voice} = \frac{\overline{T}_{call}}{\overline{T}_{call} + \overline{T}_{idle}} \times \frac{\overline{T}_{talkspurt}}{T_{interval} \times (\overline{T}_{talkspurt} + \overline{T}_{silence})} \tag{2}$$

where the first part is caused by the alternation of voice call and idle states, and the last part is given in equation (1).

Let N be the number of stations on the network and M be the number of voice calls per station, then total voice throughput on the network is

$$\lambda_{voice} \times M \times N \tag{3}$$

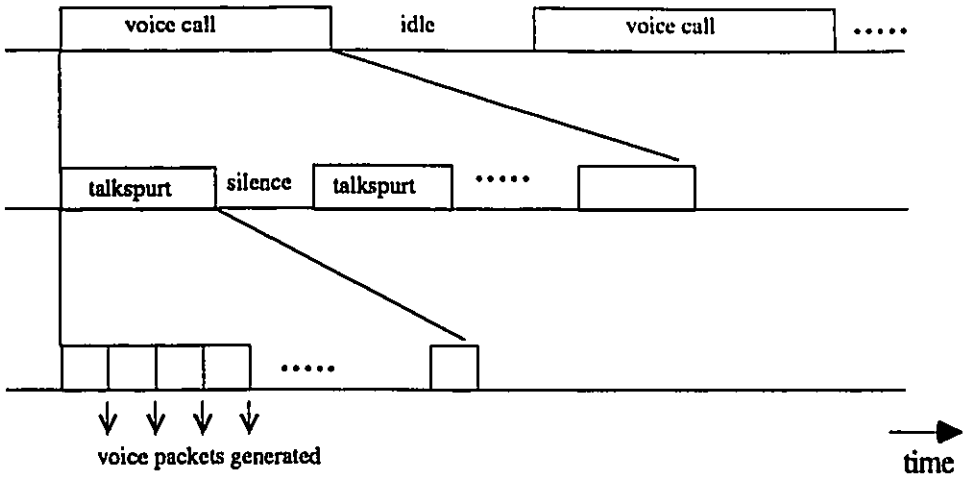


Fig. 4.1 Voice Traffic Characteristics

On the other hand, data message arrivals are assumed to be bursty with an independent Poisson arrival rate $\overline{R}_{message}$ measured in messages per time unit per station. If each message has an exponential distribution and is packetized before being transmitted, then data messages can be modeled as geometrically distributed batch arrivals with a mean message length (packets per message) of $\overline{L}_{message}$. This scenario is depicted in Fig. 4.2.

The arrival rate of data packets per station λ_{data} is

$$\lambda_{data} = \overline{R}_{message} \times \overline{L}_{message} \tag{4}$$

Assume N is the number of the stations on the network, then total data throughput (the offered data load) on the network is

$$\lambda_{data} \times N \tag{5}$$

Total throughput on the network is the sum of voice throughput and data throughput which are given in equations (3) and (5), respectively:

$$\lambda_{voice} \times M \times N + \lambda_{data} \times N \tag{6}$$

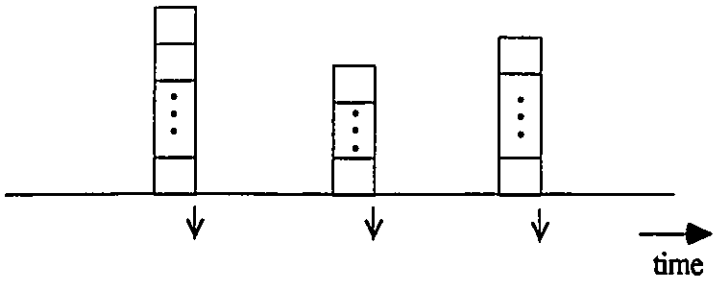


Fig. 4.2 Data Message Arrivals

4.2. Buffering Schemes

As stated in Chapter 2, voice and data traffic are packetized into packets with the same pre-defined length, and then stored in the buffer, await transmission. Assume that voice and data traffics are using separated buffer pools. We will talk about the buffering scheme for both voice and data packets.

4.2.1. Buffering Data Packets

One of the basic data traffic requirements is that there be no lost packets. When waiting for transmission, all the data packets should stay in the buffer pool. Data traffic also has the bursty characteristics. Consider all of those, the buffer sizes for data packets should be large enough. Therefore, infinite data buffer at both the origin and the destination will be assumed so that there is no data packet loss at all.

4.2.2. Buffering Voice Packets

If looking at the voice call packetization procedures in section 2.3, we know that there are buffers at both the source and the destination. Non-silent voice packets are stored in the buffer, and are waiting for transmission. Because the packet generation cycle is independent of the packet transmission process, the queue size at the buffer may continue to grow. In general, the packets can share a common buffer pool, or have a private buffer

pool for a certain source. This is what we call a sharing buffer or individual buffer. If the buffer size is so large that there is no packet loss at all, we say this buffer has infinite size. Although in reality, the buffer size is never infinite, and the packet loss is always possible. But in some circumstances, an infinite buffer can be a very good approximation to a very large buffer size, and it is a very useful condition for performance analysis using the theoretical method. Hence we will investigate this case as well. We shall assume that the buffer space at the destination is infinite so that there is no packet loss due to buffer overflow.

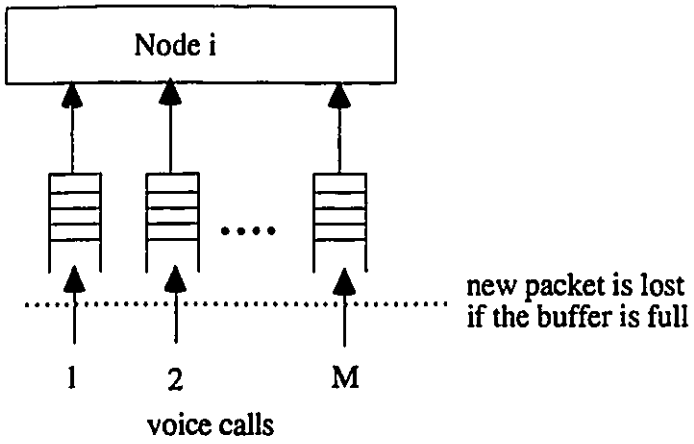
At voice source, the buffer could be shared by all voice calls arriving at this node or individually used by one voice call only. When the voice packet arrives and finds its buffer full, one packet has to be lost. In general, it is assumed that the new-comer be lost. But if note that a voice packet has a real-time requirement. Therefore, if a voice packet arrival finds a full buffer, it may be advantageous to push out the oldest packet if it has become "stale". These lead us to examine the following variations of buffering strategies:

(1) Individual Buffer and the Newest Lost (IBNL) method: each voice call has its own buffer to store at most L_1 packets, and those voice packets which arrive at the node, and find its buffer full, are lost. This is shown in Fig. 4.3(a);

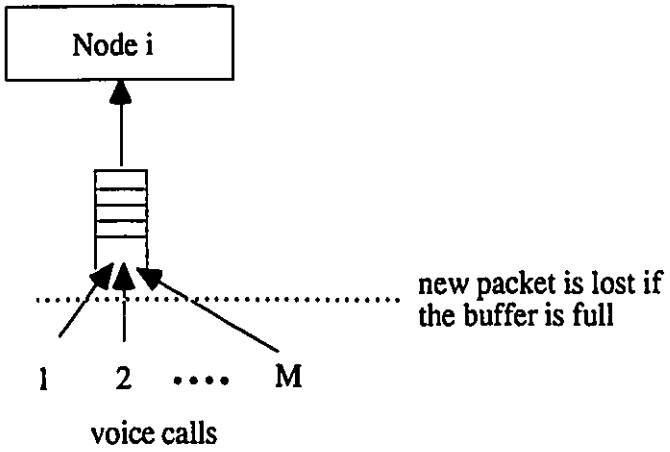
(2) Sharing Buffer and the Newest Lost (SBNL) method: one common buffer to store at most L_2 packets is shared by all voice arrivals. If a voice packet comes and finds the buffer full, this packet is lost. This is shown in Fig. 4.3(b);

(3) Individual Buffer and the Oldest Lost (IBOL) method: each voice call has its own buffer to store at most L_3 packets. The difference between this and IBNL is that, if a voice packet comes and finds its buffer full, the first packet waiting in this buffer is lost, and the new-comer will get a space at the end of the buffer. This is shown in Fig. 4.3(c).

(4) Infinite Buffer: in a special case, the voice buffer spaces at both the source and the destination are all very large so that there will be no packet loss at all. This buffering scheme is called an infinite buffer. As an example, we will use a sharing buffer at each station, as shown in Fig. 4.4.

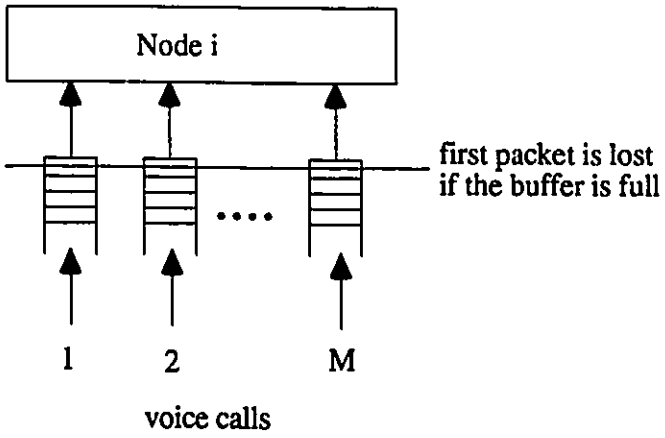


(a) Individual Buffer and the Newest Lost (IBNL)



(b) Shared Buffer and the Newest Lost (SBNL)

Fig. 4.3 Finite Buffer for Voice



(c) Individual Buffer and the Oldest Lost (IBOL)

Fig. 4.3 Finite Buffer for Voice

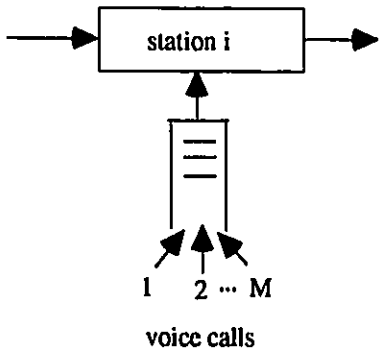


Fig. 4.4 Infinite Buffers for Voice

In the next chapter, we will have a performance evaluation of each voice buffering scheme mentioned above. We will also do some comparisons among them.

4.3. System Queueing Model

The principle of the token protocol can be modeled by the queueing system [BUX89, TAKA90]. Fig. 4.5 shows the queueing model when using our RCP. The active stations

are represented by their transmit queues. These queues are serviced in a cyclic manner symbolized by the rotating switch which represents the token. The time needed to pass the token from station i to station $(i+1)$ is modeled by a constant delay t_{walk} . On an actual ring, delay t_{walk} corresponds to the propagation delay of the signals between stations i and $(i+1)$ plus the latency caused within station i. Hence, the ring latency $T_{latency} = t_{walk} \times N$ where N is the number of stations on the network. The traffics generated in station i are characterized by the packet-arrival processes with rates $\lambda_{i,voice}$ *and* $\lambda_{i,data}$, and a packet transmission time (service time) distribution H_i with mean h_i. In this thesis, $\lambda_{i,voice} = \sum_{j=1}^{M} \lambda_j$ where λ_j is the rate from call j and j is from 1 to M, and $\lambda_{i,data}$ is the total data message rate to this station. The transmission time is the same for both voice and data packets and it is a constant denoted by one time unit P_{trans} .

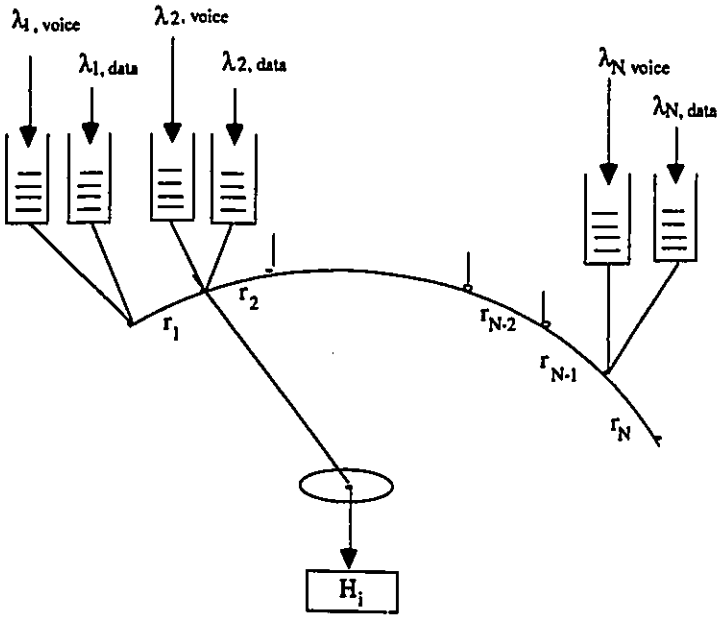


Fig. 4.5 Queueing Model of the Network

As described in Chapter 3, a typical transmission cycle, for instance, includes two token rotations when using RCP-I: the first rotation is for voice, and voice packets are eligible packets to transmit; the second rotation is for data, and data packets will be transmitted only during this time period. So the total walk time is twice that of $T_{latency}$.

In fact, the total walk time is varied in different protocols, and in different time points. The only thing we could affirm is the mean total walk time under certain conditions.

One may notice that Fig. 4.5 only shows that the voice and data traffics are using separate buffer pools, but does not include the buffering schemes. Actually, the voice buffer shown here includes all four buffering schemes in section 4.2.

4.4. QNAP2 Model

Based on all the above description, a QNAP2 model can be constructed to perform simulation. The simplified logical flow chart is shown in Fig. 4.6.

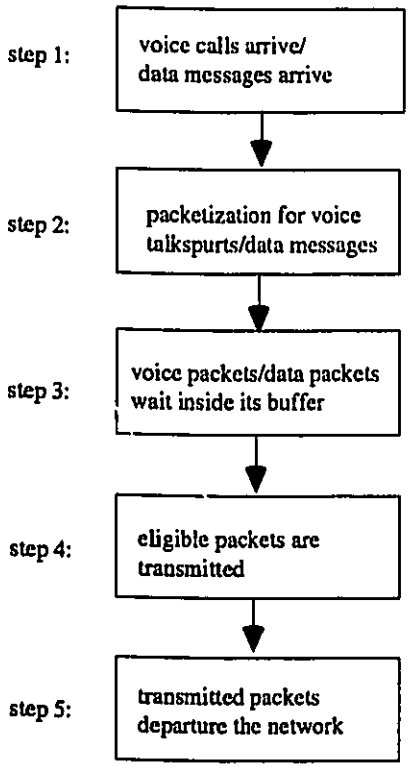


Fig. 4.6 Queueing Model of the Network

Step 1: Voice calls and data messages arrive at the station according to the distribution described in section 4.1.

Step 2: Voice talkspurts and data messages are packetized to packets as stated in section

2.3 and section 4.1. Because voice traffic is a continuous bit rate service, voice packets from each talkspurt have a constant interarrival time if a constant sample rate is employed and the packet length is predetermined. But data message arrivals are assumed to be bursty, data packets are considered as batch arrivals.

Step 3: All the voice and data packets are waiting inside its buffer pool for transmission. As stated in section 4.2, the buffer pools for data packets are large enough so that there is no data packet loss at all. But for voice packets, four different buffering schemes are employed and compared. There may be voice packet loss and this performance is measured by the voice packet loss probability.

Step 4: As shown in section 4.3, the network queueing model, the token is circulated around the network and eligible packets are transmitted. The protocol description in Chapter 3 determines who are the eligible packets and how many packets should be transmitted during the token visiting.

Step 5: The transmitted packets arrive at the destination station and leave the network. The voice delay jitters (the Coefficient of Variation of voice packet interdeparture intervals at the destination) is measured here.

From those steps, the QNAP2 code is written to simulate various network situations. The samples of the simulation codes are given in Appendix.

Up to now, we are able to carry out the performance evaluation based on the models here. In the next chapter, we will start evaluating our RCP performance.

Chapter 5

Performance Evaluation

In this chapter, we will investigate the RCP's performance. From performance results, we should have a good understanding of the behavior of the RCP, and determine how to choose some parameters.

The performance of our RCP is studied by using the QNAP2 (Queueing Network Analysis Package Two) simulation package [32]. Proper simulation times were selected to obtain convergent results by having 95% confidence intervals within ±5% of mean values and for each set of system parameters discussed below. For clarity purpose, only one figure (Fig. 5.9(a)) presents the confidence intervals.

We first define the performance measures, and then investigate the behavior and effectiveness of the two variations of the RCP serving two classes of traffic: voice and data. We also compare the performance of two variations of RCP with FDDI. We then study the case when the system is used as a backbone network, and discuss the influence of the buffering schemes on the performance.

5.1. Service Requirements and Performance Measures

Voice has two basic requirements [GRUB83, WONG84]: a small end-to-end delay for real-time delivery and a synchronous output at the destination. It can, however, tolerate a certain amount of degradation such as noise, clipping, compression, and occasional blocking without becoming objectionable. It is generally accepted that voice packet loss of less than 1 or 2 percent is tolerable [WONG89].

In local area networks, a major contributing factor to the end-to-end packet delay is the access time to the transmission medium. To satisfy the first requirement, some strategies have to be introduced. For example, FDDI uses TTRT and several timers to

control the access time, and our RCP employs the rotation counter in the token to limit the transmission cycle.

But a stream of voice packets with deterministic interarrival times to the network may not have deterministic interdeparture times at the destination, due to the variation in medium access time. This time-scale distortion can have an adverse effect on speech quality and intelligibility. One technique to reduce this distortion is destination buffering. Specifically, the first packet of each talkspurt is buffered at the destination for a delay D and then "played" out to the destination equipment. Subsequent voice packets are played out in a synchronous fashion. Due to the variation in network delay, a voice packet needed for playout at a certain time instant may be late, thus creating a "gap" or "glitch" in the output speech.

There are several factors which affect the performance of these playout schemes. One factor is the initial delay D. Other factors which affect the playout performance are the length of the talkspurt and the variation in network delay. The variation is best represented by CV (coefficient of variation of voice packet interdeparture time) at the destination. An ideal network could be characterized as a zero-jitter one (i.e., CV = 0). A more practical solution is to require jitters to have a small upper bound so that the faithful reproduction of the input voice pattern at the output can be guaranteed. For this reason, the CV is a representative factor which affects the performance of voice playout at the destination. Wong and Gopal [WONG84] have considered CV as the major concern of their study. Here we also use it as an important voice performance measure.

On the other hand, the above conditions are not usually of primary importance for data traffic. Data traffic can be categorized into two basic types: interactive and bulk. Interactive data traffic is "bursty" in nature. It usually consists of short messages requiring low network delay. Bulk data contains long messages and normally requires high throughput. However, data can in general be assumed to have higher delay tolerance at high traffic level.

Therefore, measures used in our performance evaluations are the following:

- *Voice packet delay* is defined as the duration between a voice packet is ready for transmission from a station, and the completion of its successful transmission (so that the delay is the waiting time plus the transmission time).

- *Voice delay jitters* is represented by the coefficient of variation of voice packet

interdeparture intervals at the destination (denoted by CV). In an ideal case, the mean voice packet interdeparture time should be exactly the same as the voice packet generation time, $T_{interval}$ and its variance should be zero. So the definition of CV should be:

$$CV = \frac{\sqrt{variance\ of\ voice\ packet\ interdeparture\ time}}{T_{interval}}$$

- *Voice packet loss probability* is caused by the finite buffer size for voice, and is defined as the percentage of voice packet loss.

- *Data message delay* is defined as the time duration when the first packet of this message is ready for transmission until the completion of its last packet's successful transmission.

- *Normalized network throughput* is defined as the mean number of packets generated by an application for transmission per time unit. As discussed in Chapter 2, one time unit is equal to the packet transmission time and is the same for both the voice packet and the data packet. The total network throughput is the sum of voice traffic throughput and data traffic throughput.

5.2. Assumptions

Assume voice and data traffic use separated buffer pools. Packets are placed in the buffer pools and served on a First-Come-First-Served (FCFS) basis. The buffer pool for data has infinite space so that data packets have no loss at all, voice traffic uses an infinite buffer or a finite buffer at the source so some voice packets may have loss if they do not find a sufficient amount of buffer space upon arrival.

Assume that both the voice and data traffics are using the same pre-fixed packet length so that voice packets and data packets have the same transmission time P_{trans}. The packet transmission time is assumed to be one time unit.

Although packets of the same length are used, it is conjectured that the protocol can be adapted to packets of other lengths. For example, the counter can be decremented by two for each packet that is twice as long.

Another assumption is that the network is symmetric. It means that the voice and data traffic throughput are the same at each network station.

Other assumptions include (a) a cycle length that is greater than the ring latency, (b) a constant ring latency, (c) very rare transmission errors so that no retransmissions are required from the source stations, and (d) constant reassembly and propagation delays.

For threshold parameters related to system operation, let the transmission cycle length be not greater than the voice packet generation time so that, at most, one voice packet per voice call is generated per transmission cycle. Because one station has M voice calls, it is allowed to transmit, at most, M voice packets when visited by a token during the voice round. Multiple data packets may be transmitted during the data round. Both types of transmission are limited by CIV.

As can be construed from the discussion of the RCP operations, there are quite a few parameters that would affect the performance of the two variations of RCP. We shall fix some of these parameters related to the traffic characteristics. These include the mean data message length, $\overline{L}_{message}$; the ring latency, $T_{latency}$; the mean silence period, $\overline{T}_{silence}$; the mean talkspurt duration, $\overline{T}_{talkspurt}$; the mean voice call holding time, $\overline{T}_{call}$, and the mean idle state duration, $\overline{T}_{idle}$. Their specific values are given in the table below. Note that all time values used in our study have been normalized with respect to the packet transmission time, P_{trans}.

On the other hand, the values of the idle duration of voice calls $\overline{T}_{idle}$, and the mean silence period $\overline{T}_{silence}$ are allowed to vary to effect different voice call activities on the networks. Finally, the performance is evaluated with respect to the network throughput and the threshold values, CIV.

Unless otherwise specified, the parameter values in Table 1 will be used throughout our simulation.

Table 1. Parameters used in simulation

P_{trans}	N	M	$T_{latency}$	$\overline{T}_{silence}$	$\overline{T}_{talkspurt}$	$\overline{T}_{call}$	$\overline{T}_{idle}$	$\overline{L}_{message}$
1	*10*	*1 (or 4)*	*0.005N*	*180*	*225*	*4500*	*4500*	*10*

Using the above assumptions and performance measures, the network performance is evaluated in the next section.

5.3. Simulation Results

Based on the assumptions above, the network performance in terms of the measures defined in section 5.1 are evaluated in this section. We will evaluate RCP-I and RCP-II used on a simple network, then compare them with the timed-token protocol employed by FDDI. We will also apply RCP-II on a backbone network and, at the same time, discuss different buffering schemes.

5.3.1. Performance of RCP-I and RCP-II

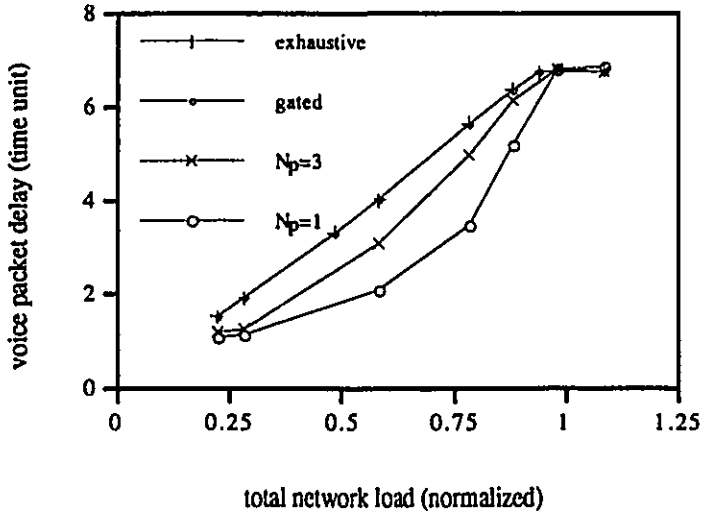
The RCP-II and RCP-II will be applied to a simple network, that means M=1 and there will be, at most, one voice call per station. Infinite buffers are used for both voice and data packets so that we will not consider packet loss at all. Under these conditions, and the parameters defined in previous section, the performance of the RCP-I and RCP-II are evaluated. In section 5.3.1.1, we will first investigate the performance of RCP-I when data traffic uses different service disciplines: exhaustive, gated, limited with a threshold $N_p = 3$, and limited with $N_p = 1$. From the obtained results, a suitable data service discipline is chosen. Then we fix the voice packet generation time and change CIV and network throughput to find out a relation between CIV and the voice packet generation time. Based on those findings, we then evaluate the performance of RCP-II: different CIV, network throughput, and different voice traffic throughput.

5.3.1.1. RCP-I

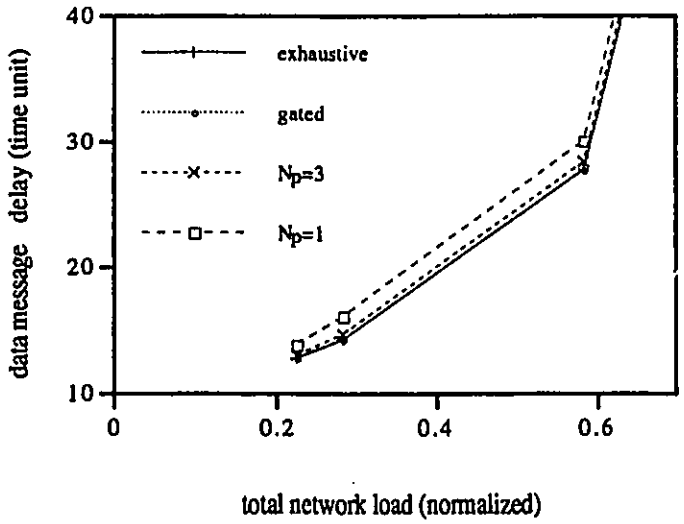
Fig. 5.1 considers a system with *N=10* stations and *CIV=10* packets. It compares the mean delays versus the total network load under four different data service disciplines: exhaustive, gated, limited with a threshold $N_p = 3$, and limited with $N_p = 1$. The mean data delay approaches a very large value as the total arrival rates from all stations gets close to the system capacity.

The mean voice delay is also affected by the disciplines as shown in Fig. 5.1(a). However, its trend is the opposite of data delay. The delays are almost equal for exhaustive and gated disciplines because the number of packets transmitted during token visits are also

limited by CIV.



(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

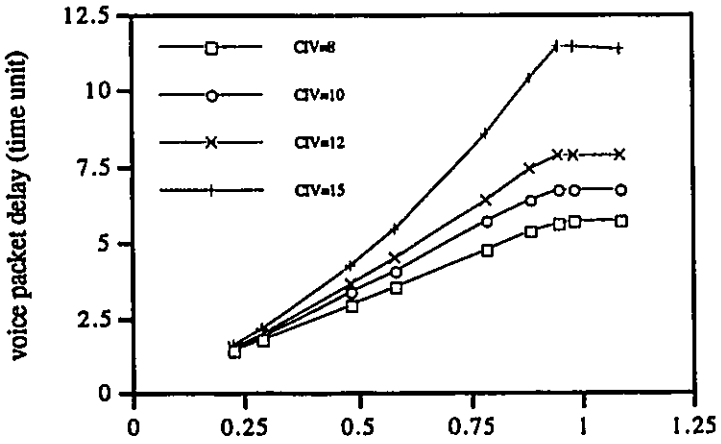
Fig. 5.1 Performance of RCP-I under Different Service Disciplines

Although a limited service with $N_p = 1$ would achieve a smaller voice packet delay at low data rates, we note that all voice delays are capped at a maximum value as the total network load increases, and there is no big difference among various disciplines. It also indicates that the delay becomes independent on the offered data traffic load. This is an

important observation because it means real-time applications can be implemented so long as we keep the capped value below the maximum threshold tolerated by voice packets. Finally, since an exhaustive service discipline gives the better channel efficiency, the exhaustive service discipline for data transmission will be used in the rest of simulation[3].

Fig. 5.2 shows how the mean delay performances vary with CIV when the system size is fixed at N=10 stations and voice packet generation time $T_{interval}$=15 time units. One notices from Figs. 5.2(a) and 5.2(b) that increasing CIV has produced lower data delay and higher voice delay. This is because for the same number of stations at the same voice load, increasing CIV allows more data on the network per cycle, and this translates to more time is required to serve the data packets.

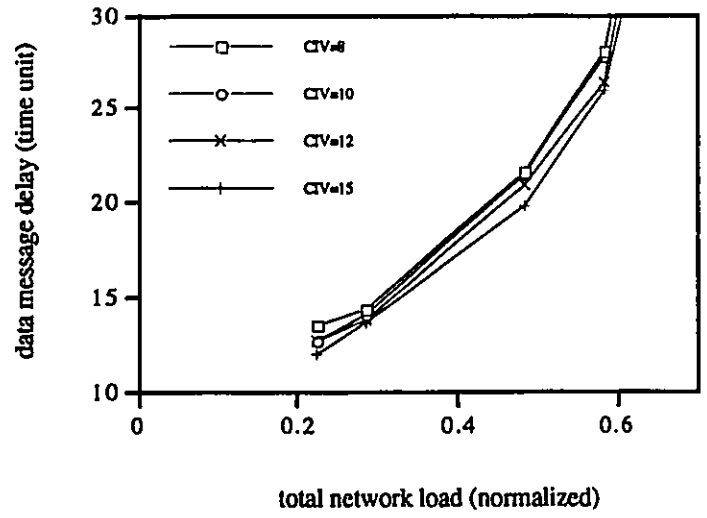
Fig. 5.2(c) shows us a very interesting phenomena: in general, the CV is increased as the CIV increases. Observe that in the CIV=15 curve, voice packet CV peaks and then drops when the network load becomes heavier. The reason is that the transmission cycle is close to 15 time units when the CIV is 15 packets and the network is overloaded. Also remember that the voice packet generation time $T_{interval}$ is 15 time units. Therefore, CIV is equal to $T_{interval}/P_{trans}$ and approximately equal to L_{cycle}/P_{trans}. This observation gives us an indication to optimize adaptively the mean voice delay and delay jitters performance accordingly.



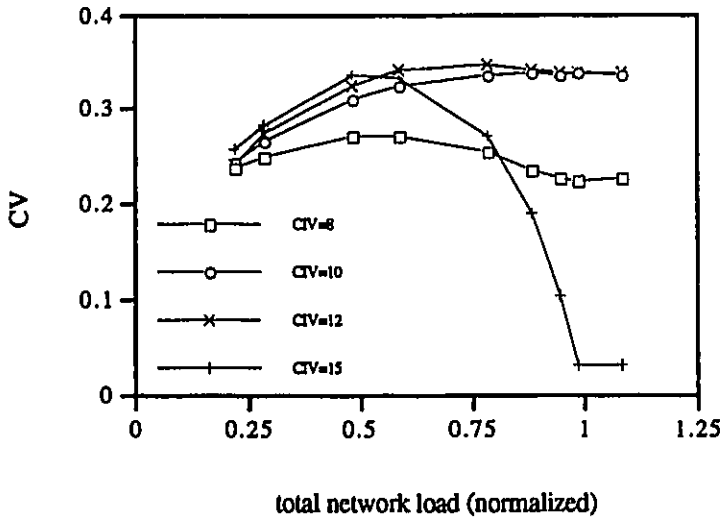
(a) Mean Voice Packet Delay

Fig. 5.2 Performance of RCP-I under Different CIV and Network Load

3 It is used in the timed token protocol as well.



(b) Mean Data Message Delay



(c) Voice Packet Delay Jitters (CV)

Fig. 5.2 Performance of RCP-I under Different CIV and Network Load (Cont'd)

5.3.1.2. RCP-II

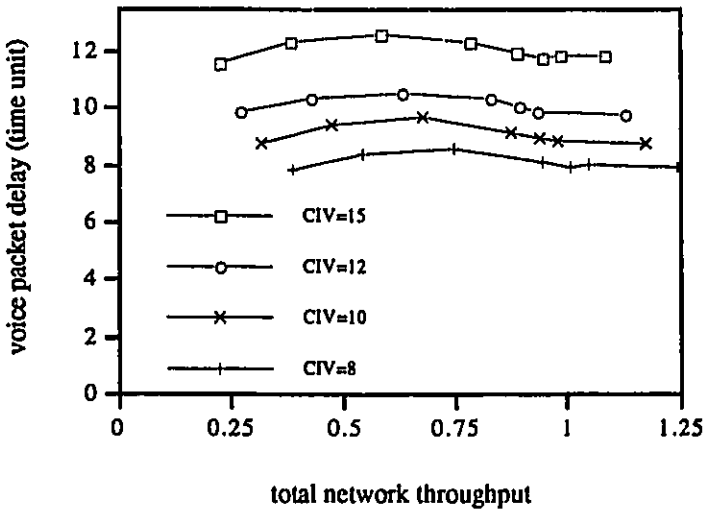
Recall that in RCP-II, one must pick a constant L_{cycle} value for the collection of packets. Various simulations have been performed with respect to L_{cycle}. Our simulation

results indicate that better voice performance in mean delay and delay jitters can be achieved when L_{cycle} is approximately equal to the packet generation time, $T_{interval}$. These results are presented in Fig. 5.3 for a network of N=10 stations and for various L_{cycle} values.

In Fig. 5.3(a), observe that for a given CIV the mean voice packet delay is essentially insensitive to the data message arrival rate, even in the overload region. This can be compared to the same performance in Fig. 5.2(a) for RCP-I where the voice delay varies with the data load. The tradeoffs are the higher voice delay at low data rates. This is not important if the delay is already below an acceptable level. Furthermore, the voice delay levels can be controlled by the proper choice of CIV as suggested in the graph.

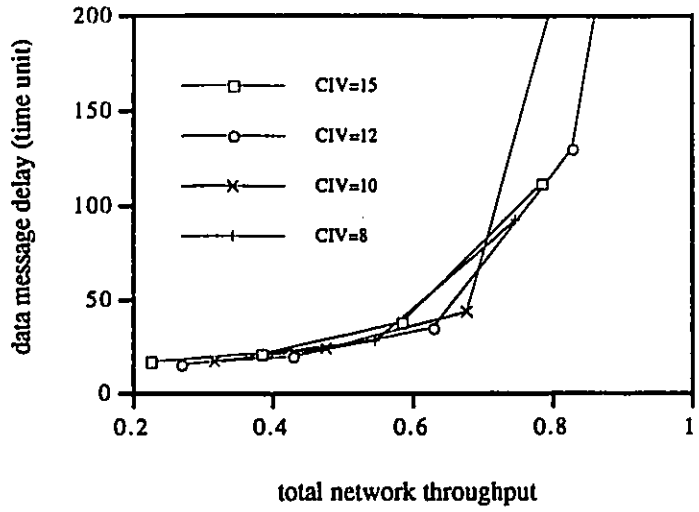
For each choice of CIV Fig. 5.3(b) indicates that there is a limit for the maximum data load allowed into the network; beyond this stability limit, the data delay increases very rapidly. In general, the smaller the CIV, the heavier the voice load and the higher the data delay. Mean data message delay mostly depends on the total network throughput, which is the sum of voice throughput and data throughput.

The delay jitters performance is presented in Fig. 5.3(c). This is obtained by measuring the interdeparture time of consecutive voice packets (i.e. those belonging to the same talkspurt). The delay jitters appear to increase slightly with respect to data load and then drops off before reaching the overload regions. Observe that the delay jitters of RCP-II is small compared with mean voice packet delay.

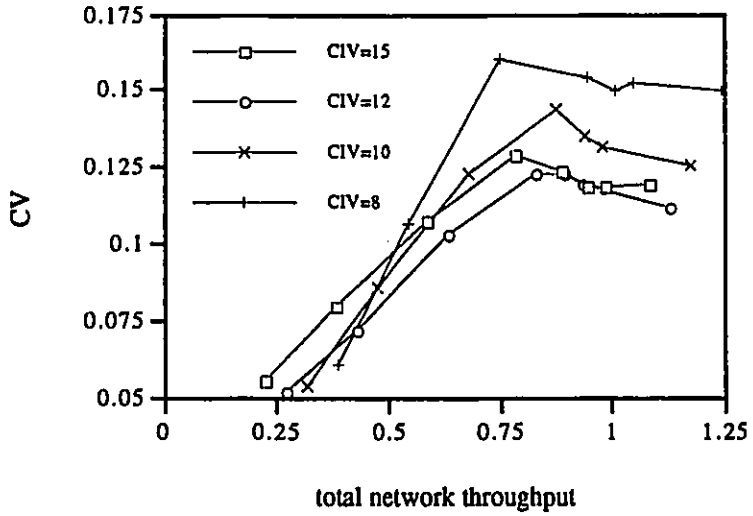


(a) Mean Voice Packet Delay

Fig. 5.3 Performance of RCP-II under Different CIV and Network Load



(b) Mean Data Message Delay

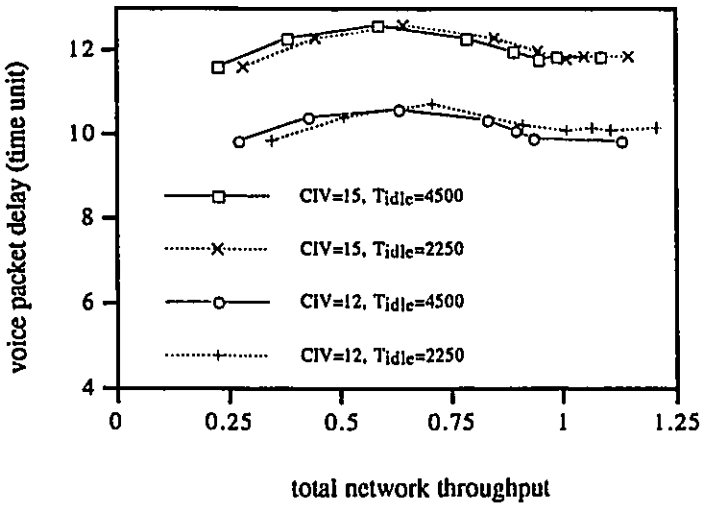


(c) Voice Delay Jitters (CV)

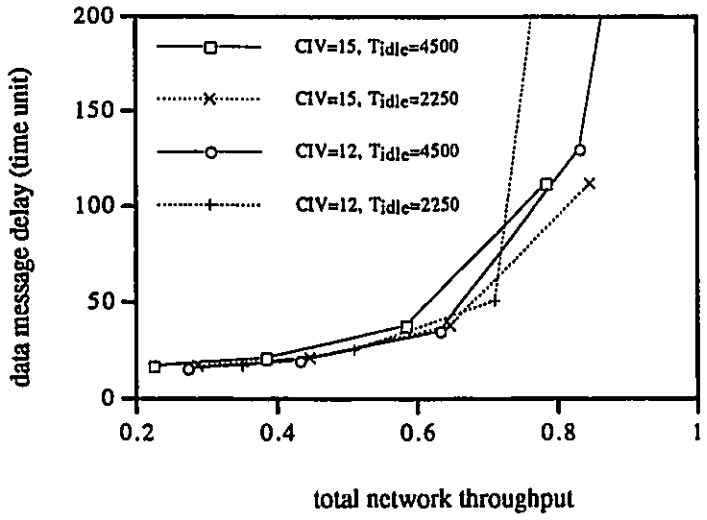
Fig. 5.3 Performance of RCP-II under Different CIV and Network Load (Cont'd)

Fig. 5.4 examines more closely the effect of voice load for a network performance. The mean voice delays are examined using CIV=12 packets, and compared for two mean interarrival times of voice calls: $\overline{T}_{idle}$ =2250 and 4500 time units. Observe that the delay performances for different voice loads are almost the same. The same observation is obtained using CIV=15 packets. This is quite interesting in contrast with the RCP-I

performance because the mean voice delay has become insensitive to both the data load and the voice load. From Fig. 5.4(b), we notice again that the mean data message delay depends on the total network throughput. Of course, the maximum allowed data load is reduced as the voice load increases (i.e., $\overline{T}_{idle}$ decreases), because the system load is the sum of the offered voice and data loads. Fig. 5.4(c) shows that the delay jitters can increase with respect to the voice load.

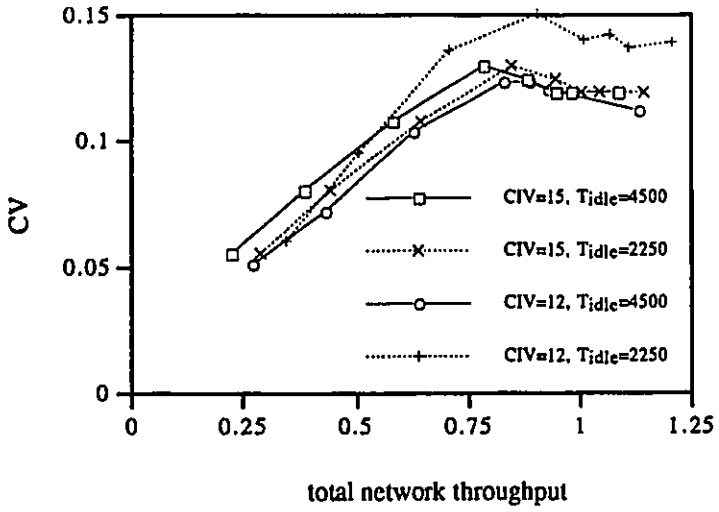


(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

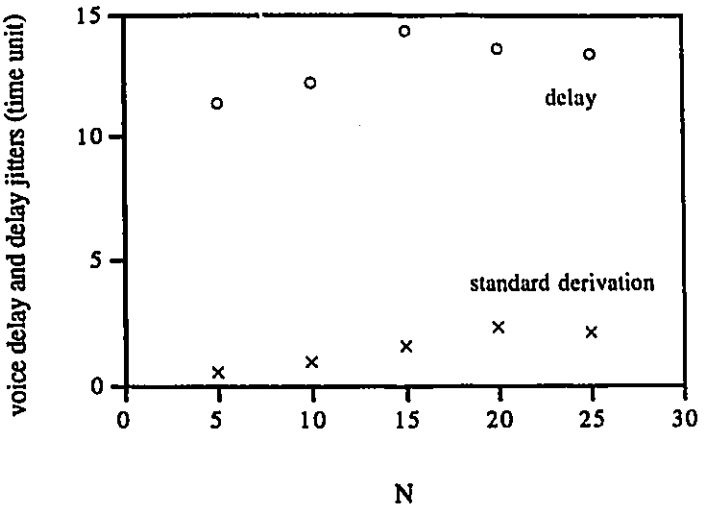
Fig. 5.4 Performance of RCP-II under Different Voice Load and CIV: $\overline{T}_{idle}$ =2250. 4500 time units and CIV=12, 15 packets



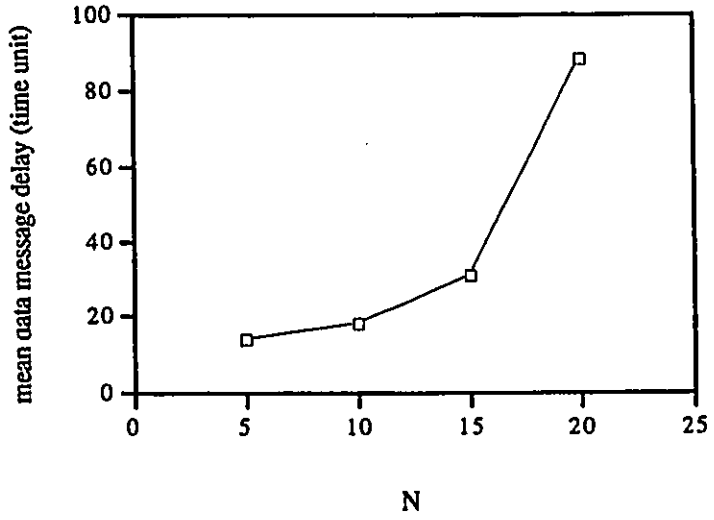
(c) Voice Delay Jitters (CV)

Fig. 5.4 Performance of RCP-II under Different Voice Load and CIV: $\bar{T}_{idle}$ =2250, 4500 time units and CIV=12, 15 packets (Cont'd)

Fig. 5.5 investigates further whether there is a dependency on the number of stations. Systems with N=5, 10, 15, 20, and 25 stations are used. Simulations are run with CIV=15 packets, $\bar{T}_{idle}$=4500 time units and a station data rate of 0.02 messages per time unit. The results appear to indicate the dependency is not strong. As shown in Fig. 5.5(a), the voice delay jitters (represent by the standard derivation of voice packet interdeparture interval) is small as well. The data delay varies with N because the total data offered load to the network is proportional to N (Fig. 5.5(b)).



(a) Voice Delay and Delay Jitters

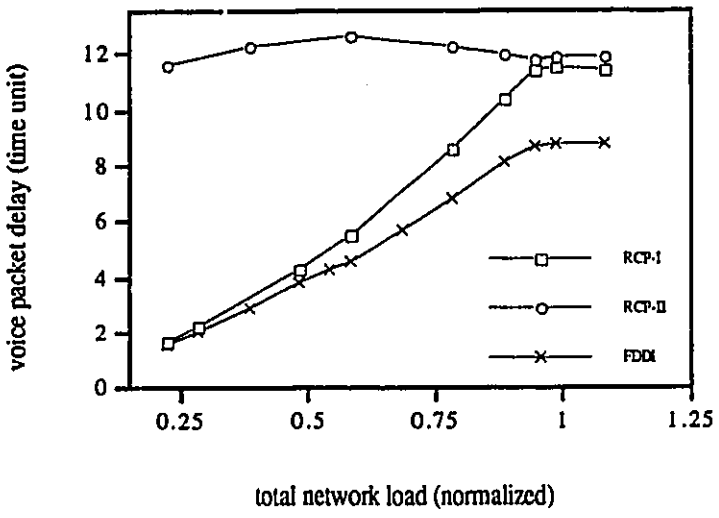


(b) Mean Data Message Delay

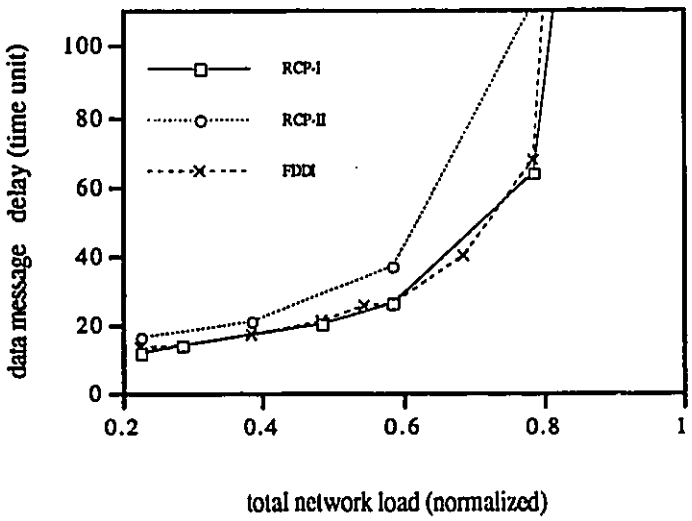
Fig. 5.5 Performance of RCP-II under Different Network Size N

5.3.2. Performance Comparison of RCP with FDDI

In Chapter 3, we compare the difference of RCP with the timed-token protocol employed in FDDI on the protocol design. The differences give rise to some interesting comparisons between the delay performances of these two protocols. The following diagrams compare the performance of FDDI timed-token protocol, the RCP-I, and the RCP-II.

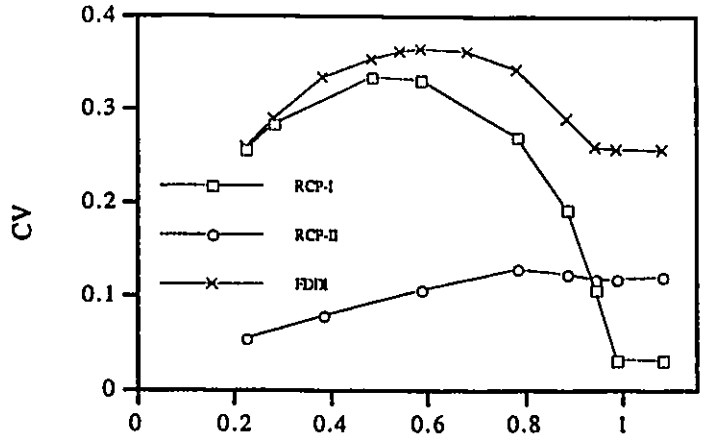


(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

Fig. 5.6 Performance Comparison of FDDI with RCP-I and RCP-II:
$\bar{T}_{idle}$ = 4500 time units, $T_{interval}$ = 15 time units, TTRT = 15 time units, CIV = 15 packets, N = 10 stations



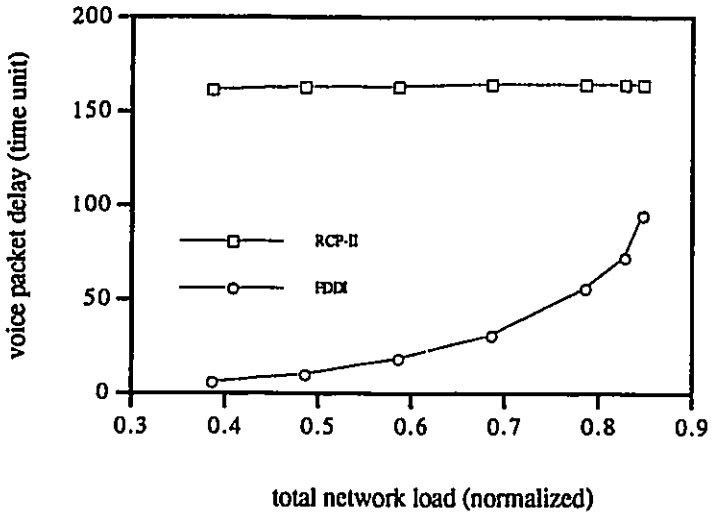
(c) Voice Delay Jitters (CV)

Fig. 5.6 Performance Comparison of FDDI with RCP-I and RCP-II: $\bar{T}_{idle}$ = 4500 time units, $T_{interval}$ = 15 time units, TTRT = 15 time units, CIV = 15 packets, N = 10 stations (Cont'd)

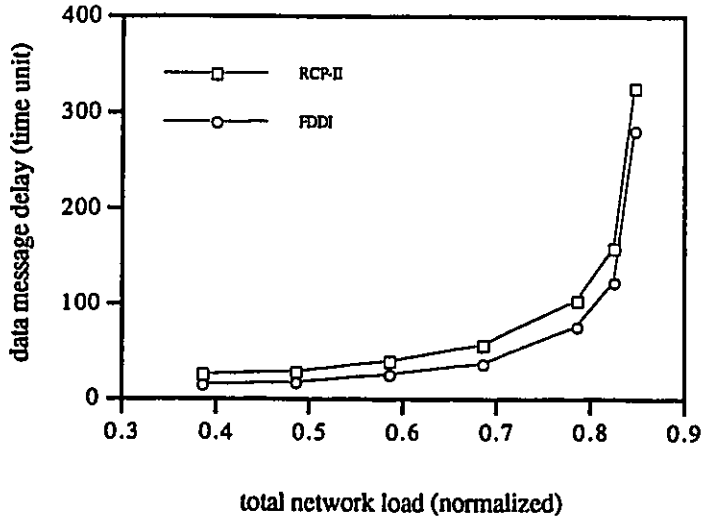
Fig. 5.6 compares the voice delay performance among an RCP-I, an RCP-II, and the FDDI protocol when they are implemented on a network of N=10 stations with $T_{interval}$ =15 time units and $\bar{T}_{idle}$ = 4500 time units. Furthermore, L_{cycle}=15 time units is used for the RCP-II, CIV=15 packets is used for the RCP-I, and TTRT=15 time units is used for the FDDI so that they have comparable token rotation time. Fig. 5.6(a) shows that the mean voice delay of FDDI increases with respect to data load, much the same as RCP-I performance. Although the almost-constant delay in RCP-II is achieved at the expense of a longer mean voice delay, the RCP-II also gives a delay jitters that is much smaller than that of the FDDI (Fig. 5.6(c)). The delay jitters of the FDDI is about 3 to 4 times larger than that of the RCP-II over the range of data load considered. It is more sensitive to the change in variations of data load. Hence, RCP-II has a much better performance for voice traffic under the similar data performance.

At the point where a station has just spent all the remaining time, and then released the token, the downstream station on the FDDI network is not able to transmit immediately, it has to wait until the token rotates one more round and then come back. If using the RCP, the downstream station starts transmission right after it captures the token. Therefore,

compared with FDDI protocol, RCP-I gives better mean data delay, but RCP-II is a little worse since it wastes some time to rotate the token without doing any transmission (fortunately, they are very close when the network load is heavy). This is shown in Fig. 5.6(b).

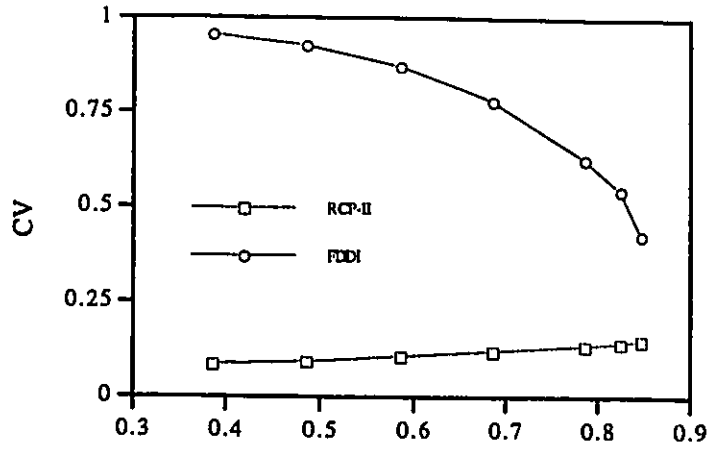


(a) Mean Voice Packet Delay

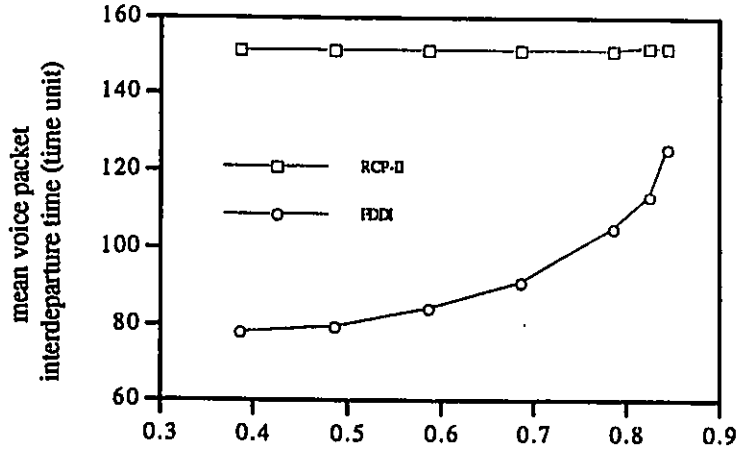


(b) Mean Data Message Delay

Fig. 5.7 Performance Comparison of RCP-II with FDDI when N=100 stations: $\bar{T}_{idle}$ = 4500 time units, $T_{interval}$ = 150 time units, TTRT = 150 time units, CIV = 150 packets



(c) Voice Delay Jitters (CV)



(d) mean voice packet interdeparture time

Fig. 5.7 Performance Comparison of RCP-II with FDDI when N=100 stations: $\bar{T}_{idle}$ = 4500 time units, $T_{interval}$ = 150 time units, TTRT = 150 time units, CIV = 150 packets (Cont'd)

Fig. 5.7 compares the performance of RCP-II with FDDI timed token protocol when the number of station on the network N=100. Here we use CIV=150 packets for RCP-II and TTRT = 150 time units for FDDI protocol. Voice packet generation time for both is 150 time units. All other parameters are the same as those used in Fig. 5.6.

The results show that when using RCP-II, although the mean voice packet delay is larger, it is independent of the network load. The mean voice packet interdeparture time is unchanged with the network load and its CV is much smaller than that using FDDI protocol. This observation is representative since it has also been found in a small size network.

This means that RCP-II will have the expected performance even for the ring with a large number of stations on it.

5.3.3. Performance of RCP-II as a Backbone Network

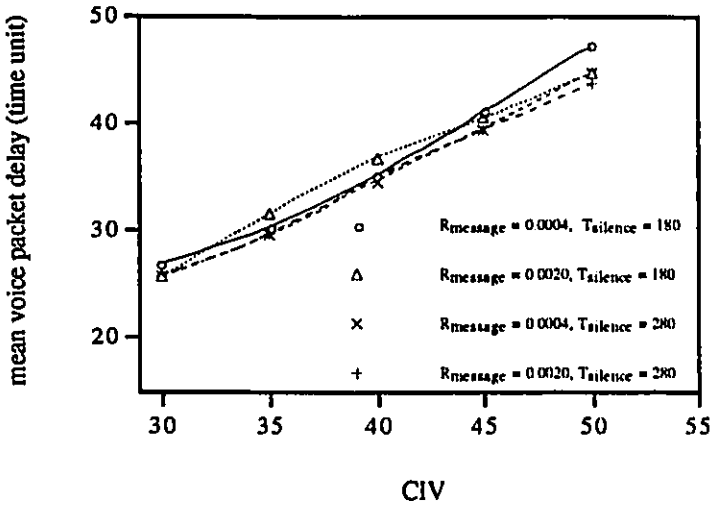
In this section, we will evaluate performance when the system is used as a backbone network. As an example, the RCP-II is employed, that means each station could perform functions such as a bridge or a concentrator. Therefore, each station may have multiple voice call arrivals and the data messages may come from different nodes which are connected to this station. For simplicity, the multiple voice calls is considered as less voice calls which are always off-hook. So we use M=4 voice calls per station, each voice call has no idle time, i.e., $\overline{T}_{idle} = 0$ or $\overline{T}_{call} \to \infty$, and by changing the mean silence duration $\overline{T}_{silence}$, the voice traffic load is changed. The multiple data arrivals are considered as the increase of the total data message arrival rate. As assumed in Chapter 4, each data arrival has an independent Poisson arrival rate, so the total data message arrival $\overline{R}_{message}$ is still Poisson distributed and its rate is the sum of each data message arrival rate [KLEI75].

We also consider different buffering schemes for voice packets which are described in section 4.2. The fourth buffering scheme, infinite buffer, will be investigated first since this is the simplest case. Then the first three buffering schemes (IBNL, SBNL, and IBOL) will be discussed. Only at this point can we measure the voice packet loss probability.

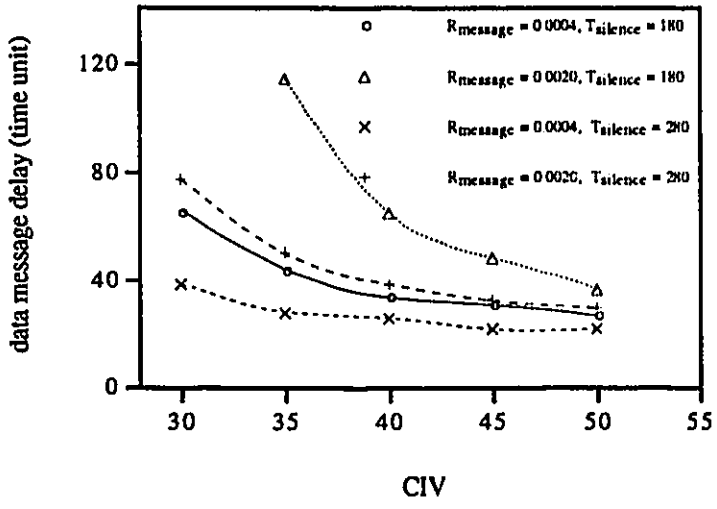
5.3.3.1. Infinite Buffer for Voice Packets

Fig. 5.8(a), (b), and (c) show the influence of CIV (Counter Initial Value) on the mean voice packet delay, mean data message delay, and voice delay jitters (CV), respectively. A family of 4 curves are shown for different combination of the data message arrival rate $\overline{R}_{message}$ and mean silence duration $\overline{T}_{silence}$. In general, increasing CIV results in longer voice packet delay, shorter data message delay, and smaller CV. This is because

larger CIV is equivalent to longer Transmission cycle length, then each cycle may serve more data packets. By extending the mean silence period, the voice traffic load is reduced so that the total network throughput decreased. It makes the mean voice packet delay, the data message delay, and the CV smaller.

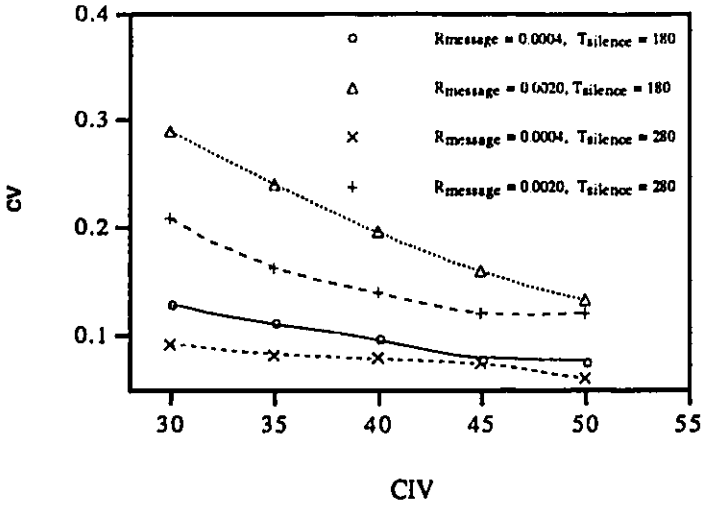


(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

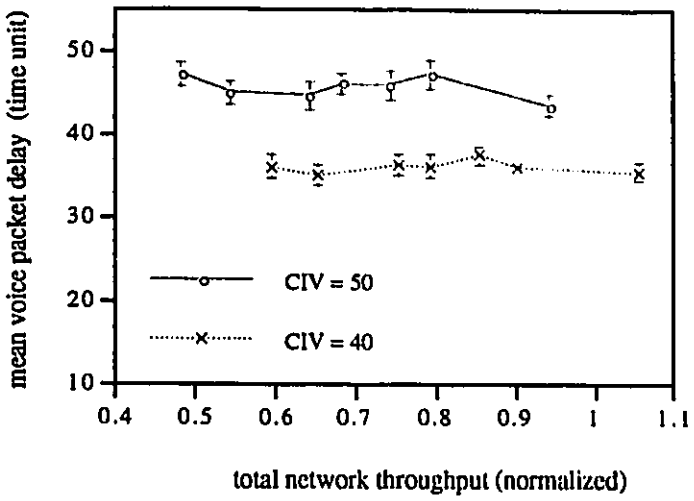
Fig. 5.8 Performance of RCP-II Using Infinite Voice Buffer versus CIV under Different Voice Load and Data Load



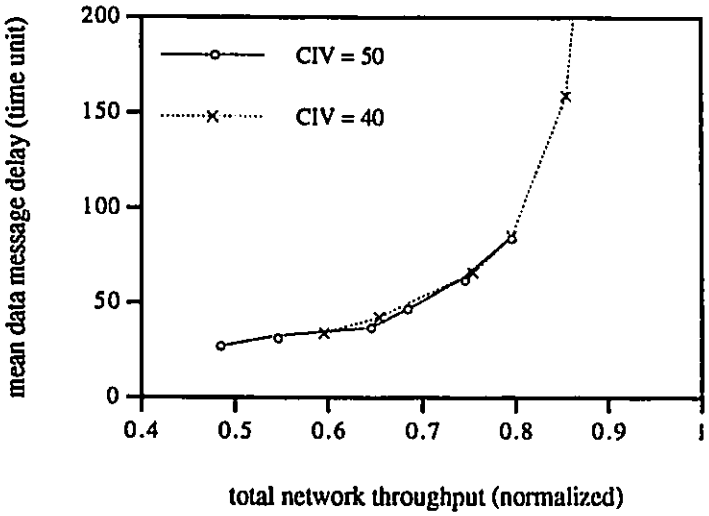
(c) Voice Delay Jitters (CV)

Fig. 5.8 Performance of RCP-II Using Infinite Voice Buffer versus CIV under Different Voice Load and Data Load (Cont'd)

Fig. 5.9(a), (b), and (c) present the influence of data load on the mean voice packet delay, mean data message delay, and voice delay jitters (CV), respectively, where $\overline{T}_{silence}$ is keeping at 180 time units. Data load is different when the data message arrival rate $\overline{R}_{message}$ is changed. For fixed CIV (40 or 50), the voice traffic load is unchanged. So, when the mean data message arrival rate becomes bigger, the data load is heavier, then the total network throughput increases. Fig. 5.9(a) tells us that the mean voice packet delay is almost independent of the throughput. It is one of our RCP's advantages. Fig. 5.9(b) indicates a very interesting phenomenon: the mean data message delay depends only on the total network throughput. Notice that the mean voice packet delay is remaining unchanged even when the total network throughput is greater than 1. This means that when the data traffic load is so heavy that the network is saturated, the voice packets can still be transmitted as required, but data packets are delayed. So we could say that the voice packet transmission is independent of the data traffic. The results are completely confirm our design prediction. Fig. 5.9(c) shows that the maximum CV is capped at a certain point.

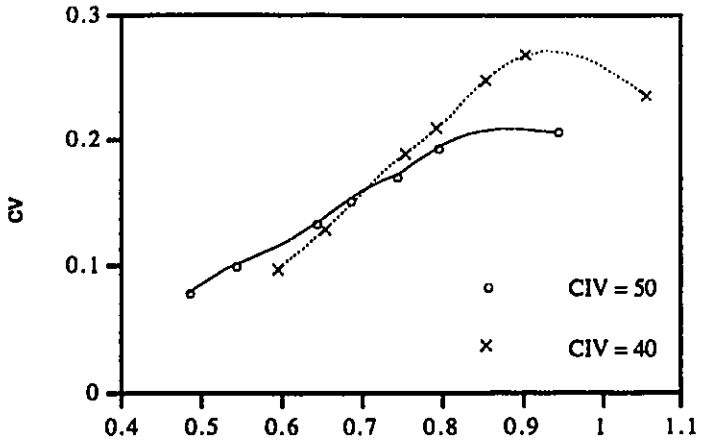


(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

Fig. 5.9 Performance of RCP-II Using Infinite Voice Buffer versus Network Load under Different CIV

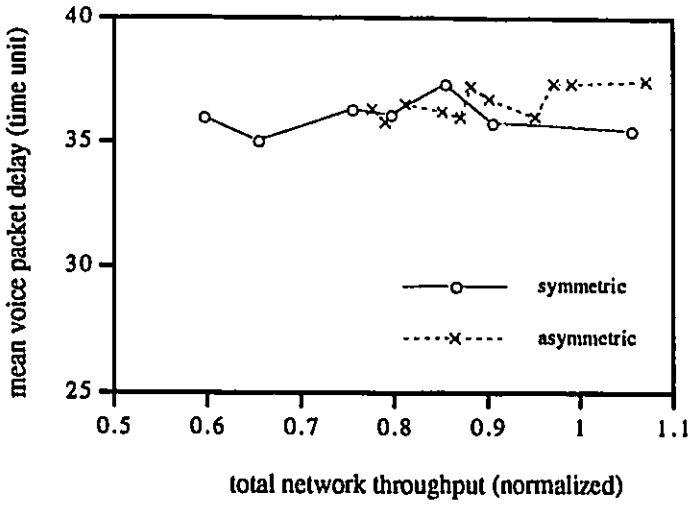


(c) Voice Delay Jitters (CV)

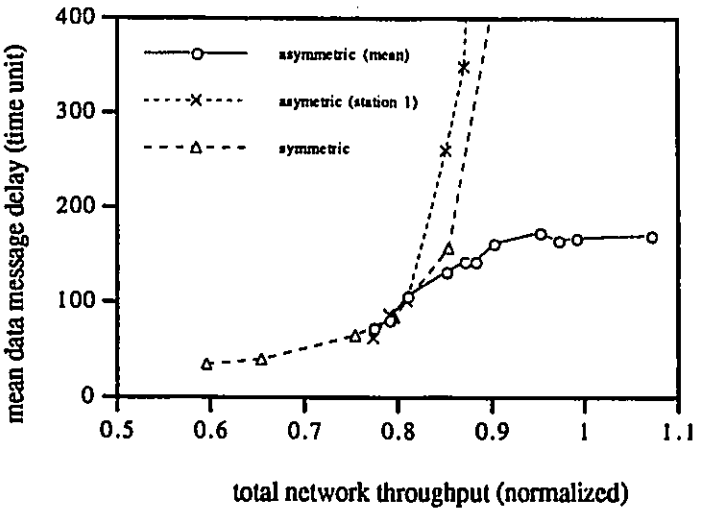
Fig. 5.9 Performance of RCP-II Using Infinite Voice Buffer versus Network Load under Different CIV (Cont'd)

Fig. 5.10 shows the network performances when the network is loaded in asymmetric situation. That is, let the data arrival rate $\overline{R}_{message}$ of the station from 2 to 10 fix at 0.0024 messages/time unit, and have $\overline{R}_{message}$ at station 1 varied. Also $\overline{T}_{silence}$ is keeping at 180 time units and CIV is fixed CIV at 40 packets. All the performances are compared with the symmetric case as well.

Because the mean voice packet delay at each station has no obvious difference, the average value from all N=10 stations is used in Fig. 5.10(a). It shows that the mean voice packet delay is similar to the symmetric case. Fig. 5.10(b) shows the data message delay of the station 1 and of the average of station 2 to 10 as well as that in the symmetric situation. It indicates that when the data arrival rate of station 1 increases, its delay grows quickly, especially when the network load is close to 1. But the data message delay at all the other stations keeps at a certain value. Fig. 5.10(c) means that the CV is affected slightly. All those results tell us that if a station has a heavy data traffic load, it does not monopolize the network. It only causes itself a very big data message delay. This shows our RCP is also working well in an asymmetric network.

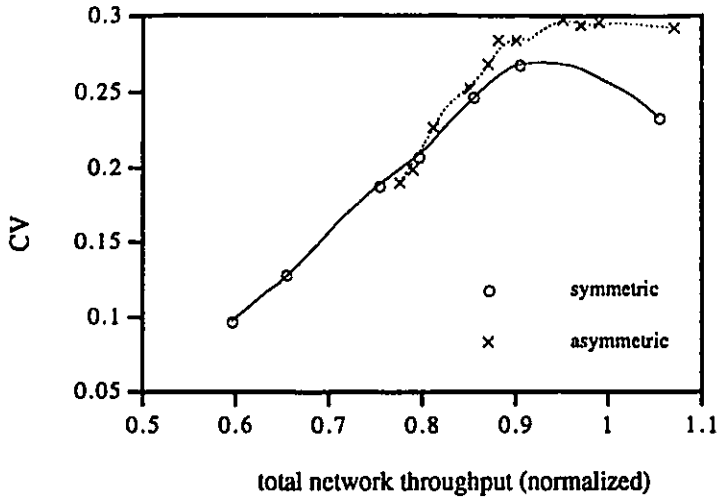


(a) Mean Voice Packet Delay



(b) Mean Data Message Delay

Fig. 5.10 Performance of RCP-II in Asymmetric Environment



(c) Voice Delay Jitters (CV)

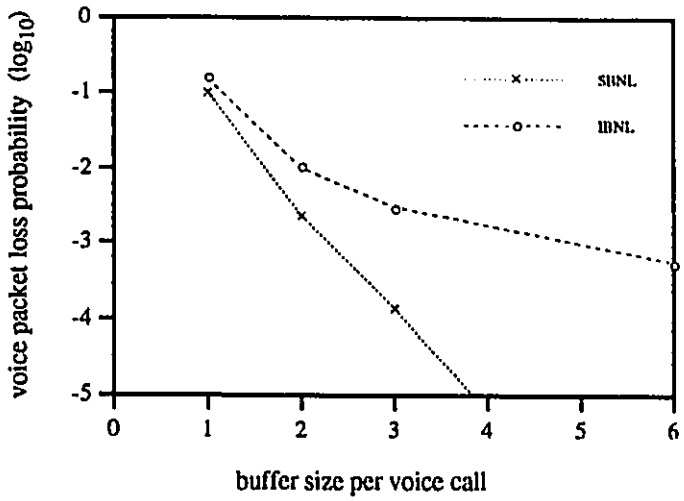
Fig. 5.10 Performance of RCP-II in Asymmetric Environment (Cont'd)

5.3.3.2. Finite Buffer for Voice Packets

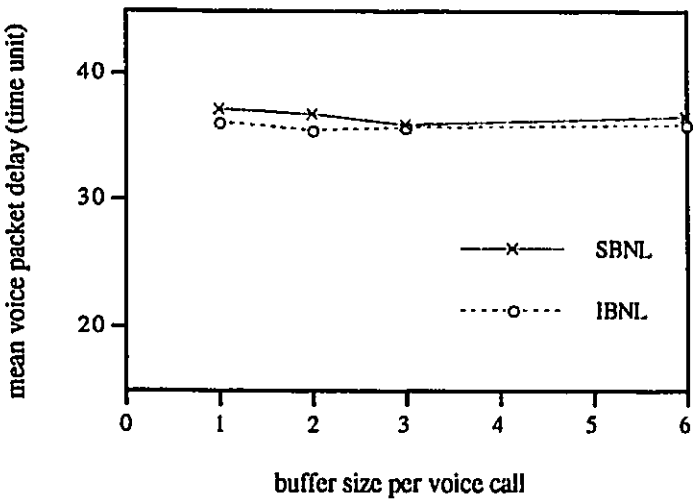
We have different buffering schemes as discussed in Chapter 4. The buffer could be shared or individually used. The buffer could also be designed in such a way that the lost packet is the newest arrival packet or the longest waiting packet (we call it the oldest lost). For comparison purpose, we are firstly to consider the cases of the shared buffer or individual buffer while having the newest packet lost, i.e., SBNL and IBNL. Then we will fix the buffer as an individual used buffer, and compare the cases of the newest packet lost and the oldest packet lost, that is, IBNL and IBOL.

First of all, we will see how the buffer size affects the performances. We fix CIV=40 packets and $\overline{T}_{silence}$ = 180 time units. At a particular buffer size, we will use the average values of the voice packet loss probabilities and the mean voice packet delay. The average values are the mean of each different data message arrival rate. The reason we use the average values is that, when the data message arrival rate varies, the voice packet loss probabilities and the mean voice packet delay almost remain unchanged.

Fig. 5.11 compares the loss probabilities and mean voice packet delays when the SBNL and IBNL buffering schemes are used.



(a) Voice Packet Loss Probability



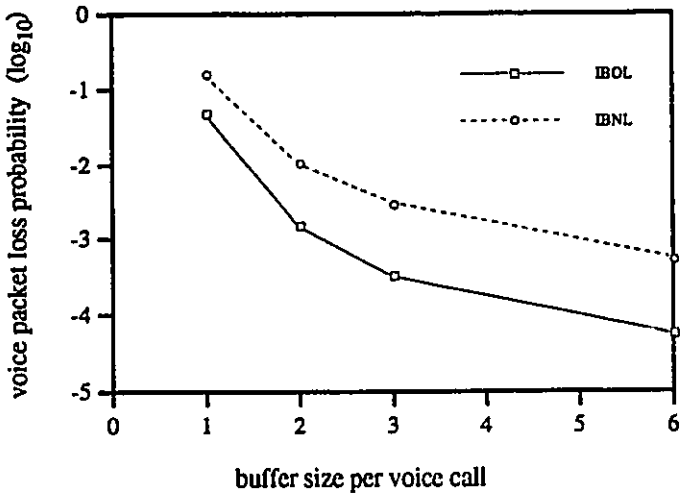
(b) Mean Voice Packet Delay

Fig. 5.11 Performance of RCP-II Using Finite Voice Buffer versus Buffer Size under SBNL and IBNL

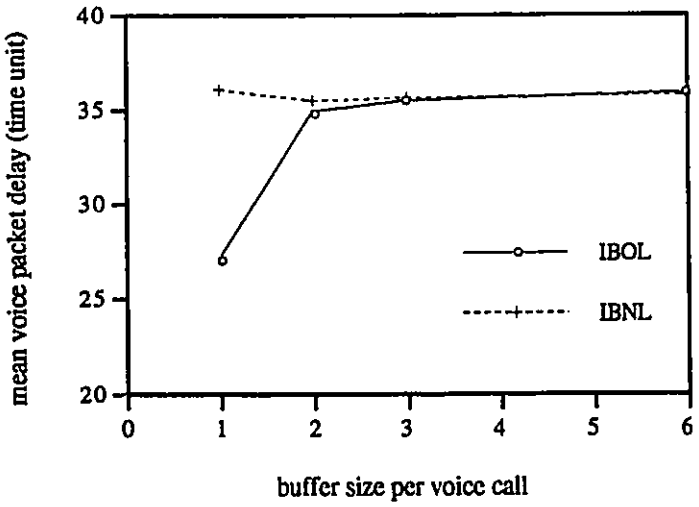
From Fig. 5.11(a), we could say that the shared buffering method always gives smaller loss probability than the individual buffering scheme does. The bigger the buffer size is, the bigger the difference is. This reflects a fact which is generally accepted. From Fig. 5.11(b), we know that the mean voice packet delays are very close but the IBNL gives

a little bit shorter delay. This may be caused by the different loss probabilities, if the voice packet loss probability is larger, the delay should be shorter because there are less packets waiting.

Fig. 5.12 compares the loss probabilities and mean voice packet delays when the IBOL and IBNL buffering schemes are used.



(a) Voice Packet Loss Probability

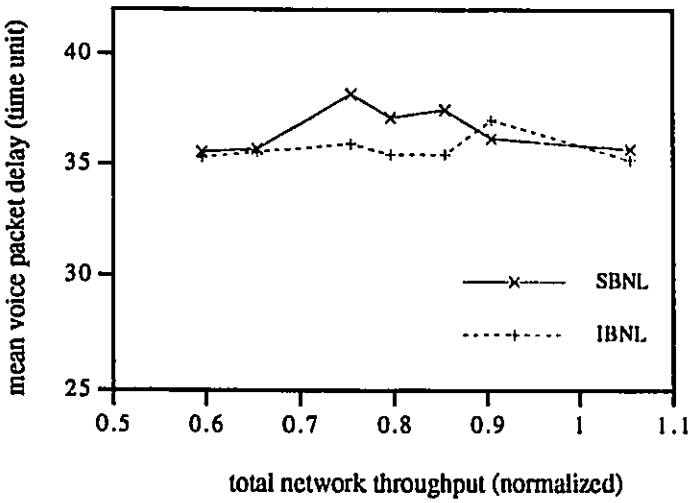


(b) Mean Voice Packet Delay

Fig. 5.12 Performance of RCP-II Using Finite Voice Buffer versus Buffer Size under IBOL and IBNL

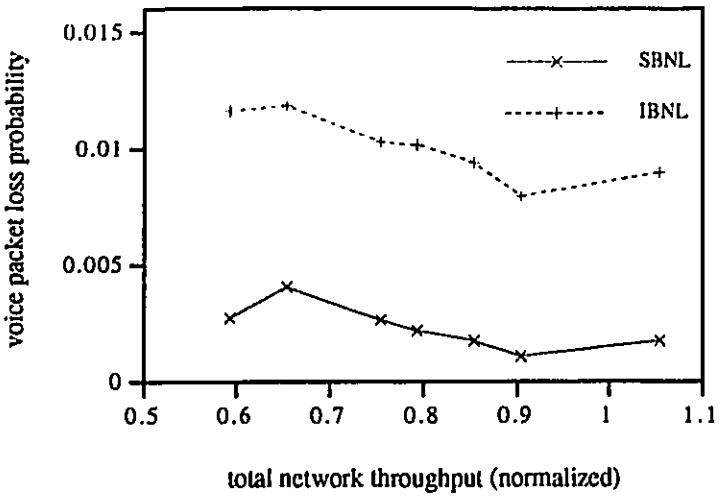
From Fig. 5.12(a), we know the IBOL and IBNL have the same tendency for the voice packet loss probability, but the IBOL's loss probability is smaller than the IBNL's. Fig. 5.12(b) indicates that the mean voice packet delay using IBOL is smaller especially at the point when using IBOL and the buffer size is 1. This is a reasonable and interesting observation: when the waiting voice packet instead of the new coming one is lost, mean voice packet waiting time is reduced. It shows us that the IBOL method is a good choice because it produces the smaller mean voice packet delay (particularly when the buffer size is small), and the lower voice packet loss probability.

Secondly, we will see how the data message arrival rate (i.e., the data traffic throughput) influences the performances. Let CIV=40 packets and $\overline{T}_{silence}$ = 180 time units. Fig. 5.13 will use the SBNL and IBOL buffering schemes and Fig. 5.14 is deploying the IBOL and IBNL buffering schemes.

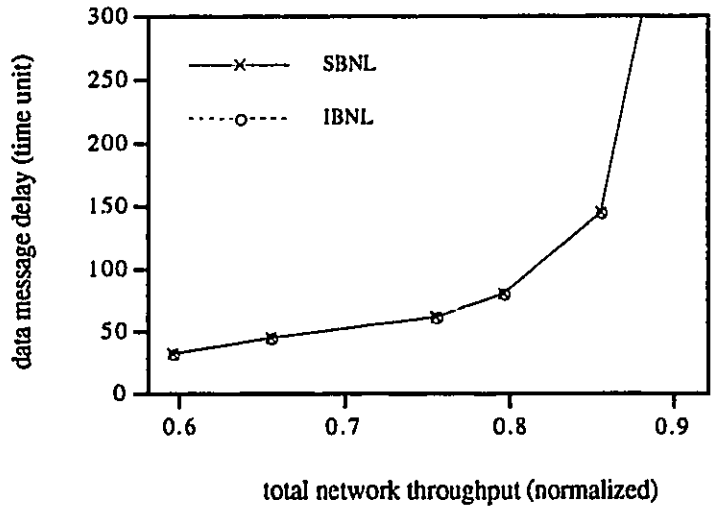


(a) Mean Voice Packet Delay

Fig. 5.13 Performance of RCP-II under Finite Voice Buffer versus Network Throughput under SBNL and IBNL

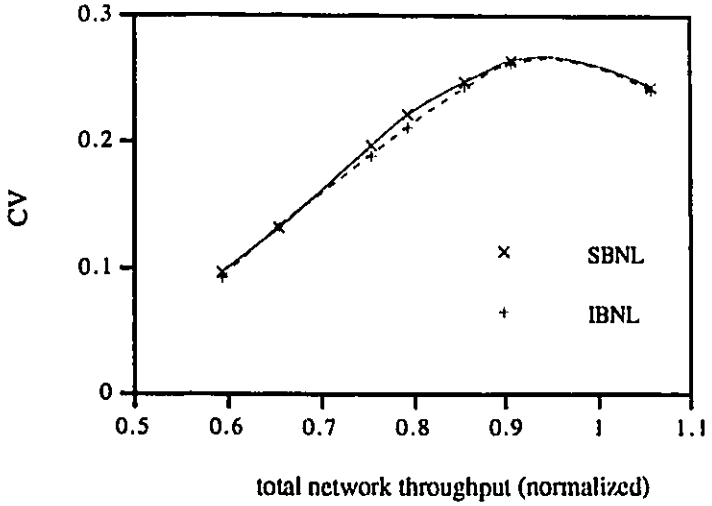


(b) Voice Packet Loss Probability



(c) Mean Data Message Delay

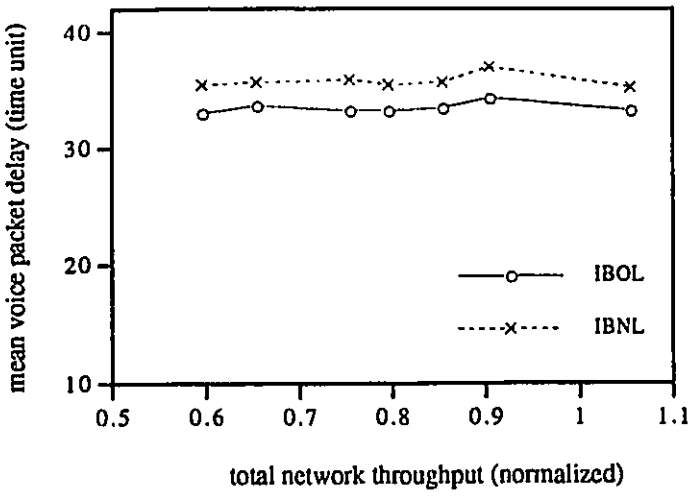
Fig. 5.13 Performance of RCP-II under Finite Voice Buffer versus Network Throughput under SBNL and IBNL(Cont'd)



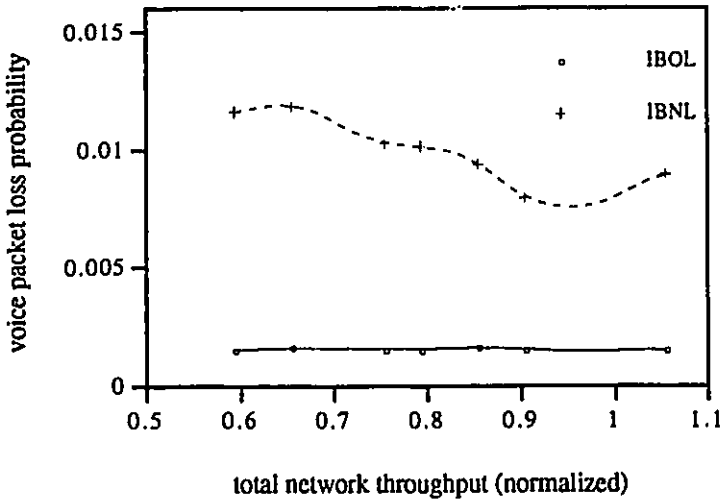
(d) Voice Delay Jitters (CV)

Fig. 5.13 Performance of RCP-II under Finite Voice Buffer versus Network Throughput under SBNL and IBNL(Cont'd)

Fig. 5.13(a) uses the average value to show the mean voice packet delay performance. At a particular data arrival rate, the average value is the mean of each different buffer size. This is because the mean voice packet delay is almost stable when the buffer size is changed. Fig. 5.13(b), (c), and (d) give the voice packet loss probability, mean data message delay, and coefficient of variation of the voice packet interdeparture time versus the total network throughput, respectively, when the buffer size per voice call is fixed at 2. Like the results shown in Fig. 5.11, we still got the almost same voice packet delay, the voice packet loss probability using SBNL is smaller than that using IBNL. The data message delay and CV roughly depend only on the total network throughput.

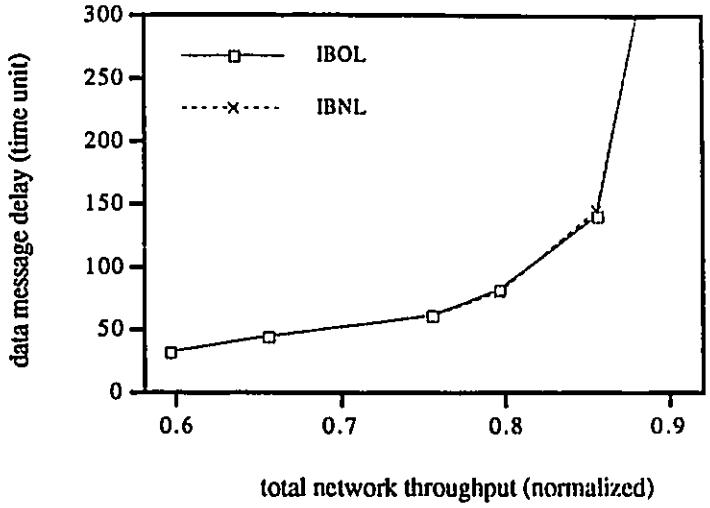


(a) Mean Voice Packet Delay

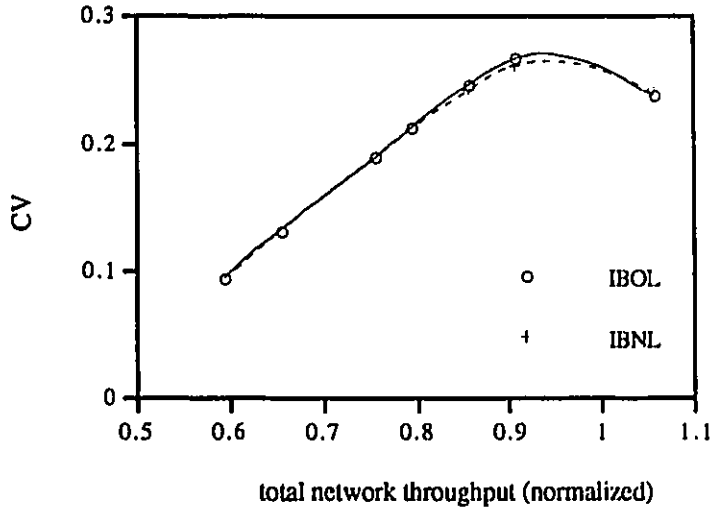


(b) Voice Packet Loss Probability

Fig. 5.14 Performance of RCP-II under Finite Voice Buffer versus Network Throughput under IBOL and IBNL



(c) Mean Data Message Delay



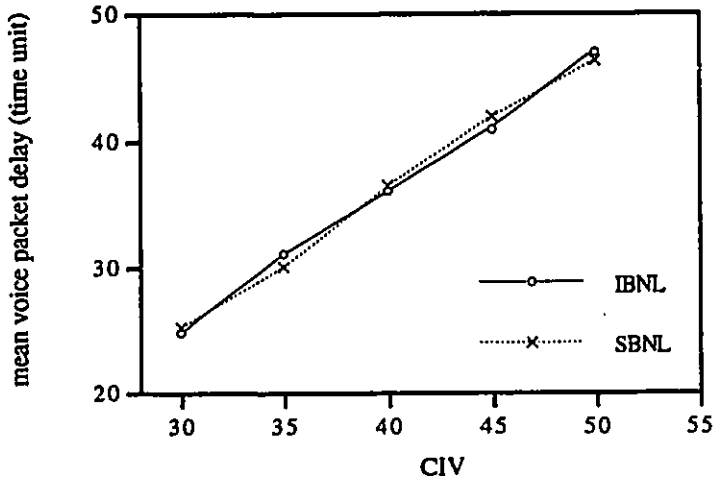
(d) Voice Delay Jitters (CV)

Fig. 5.14 Performance of RCP-II under Finite Voice Buffer versus Network Throughput under IBOL and IBNL(Cont'd)

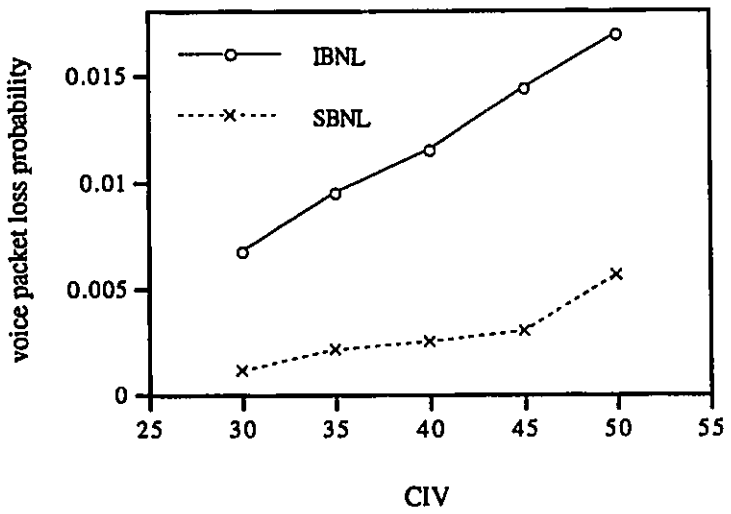
Under the same conditions as Fig. 5.13 but changing the buffering schemes to IBOL and IBNL, Fig. 5.14(a), (b), (c), and (d) show the mean voice packet delay, voice packet loss probability, mean data message delay, and voice delay jitters (CV), respectively. Like Fig. 5.12, the mean voice packet delay using IBOL is smaller than that using IBNL, and

the voice packet loss probability using IBOL is also lower. Similar to Fig. 5.13, the data message delay and CV are dependent on the total network throughput but the buffering schemes.

Thirdly, we would like to see the influence of varied CIV on the performance. Assume the buffer size per voice call is fixed at 2, the data message arrival rate is 0.0004 messages/time unit per station, and $\overline{T}_{silence}$ is 180 time units. Fig. 5.15 use the SBNL and IBNL buffering schemes and Fig. 5.16 deploys the IBOL and IBNL buffering schemes.

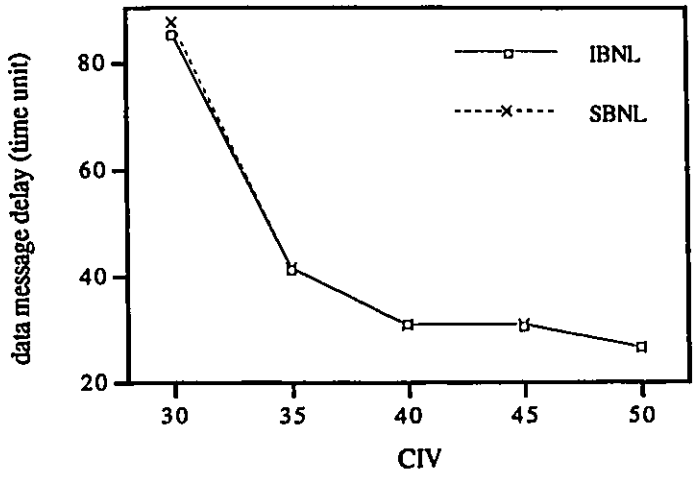


(a) Mean Voice Packet Delay

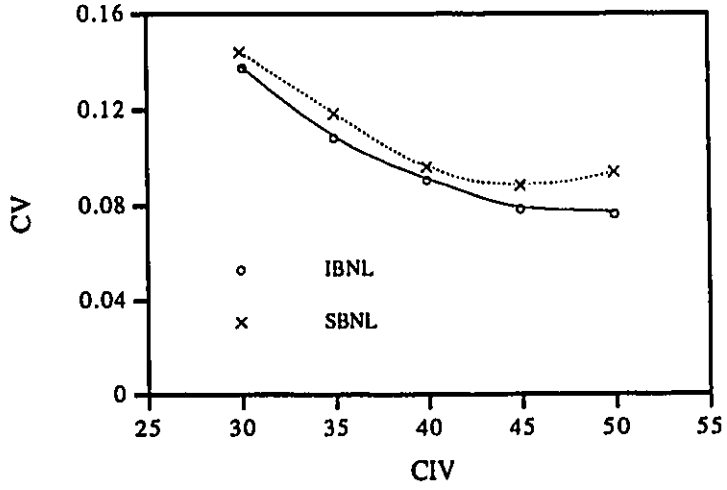


(b) Voice Packet Loss Probability

Fig. 5.15 Performance of RCP-II Using Finite Voice Buffer versus CIV under SBNL and IBNL



(c) Mean Data Message Delay

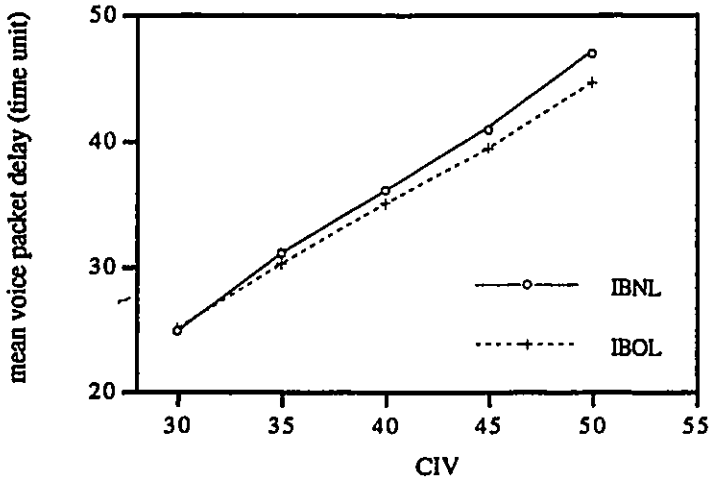


(d) Voice Delay Jitters (CV)

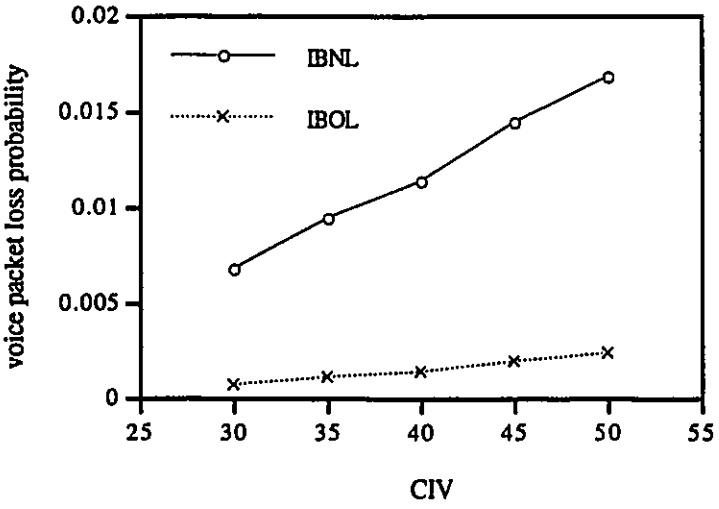
Fig. 5.15 Performance of RCP-II Using Finite Voice Buffer versus CIV under SBNL and IBNL

Fig. 5.15(a), (b), (c), and (d) show the influence of CIV on the mean voice packet delay, voice packet loss probability, mean data message delay, and voice delay jitters under the SBNL and the IBNL buffering methods, respectively. A larger CIV is equivalent to longer transmission cycle length so that each cycle can serve more data packets. Therefore, increasing CIV results in a longer voice packet delay and a shorter data message delay. Also, the longer transmission cycle causes longer length of voice packets so that a certain

voice talkspurt period is packetized in a lesser number of voice packets. But a voice packet has the same chance to be lost. Therefore, a larger voice packet loss probability is concluded. The same as Fig. 5.11 again, for buffering methods SBNL and IBNL, we still obtain almost the same voice packet delay, the voice packet loss probability using SBNL is smaller than that using IBNL. But the CV using SBNL is larger than that using IBNL. This may be a tradeoff between SBNL and IBNL.

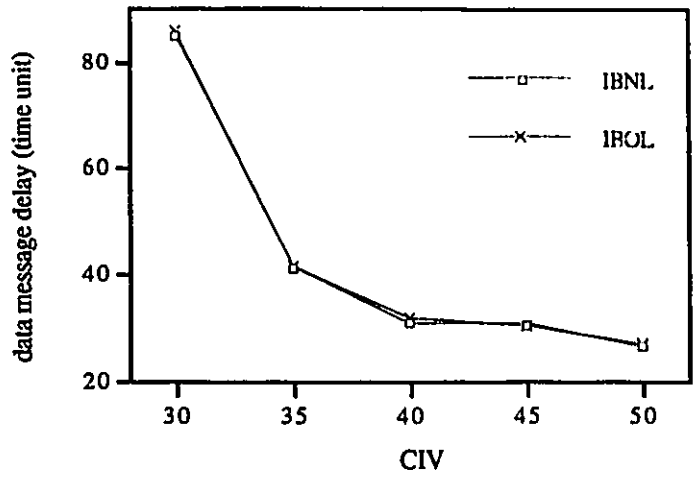


(a) Mean Voice Packet Delay

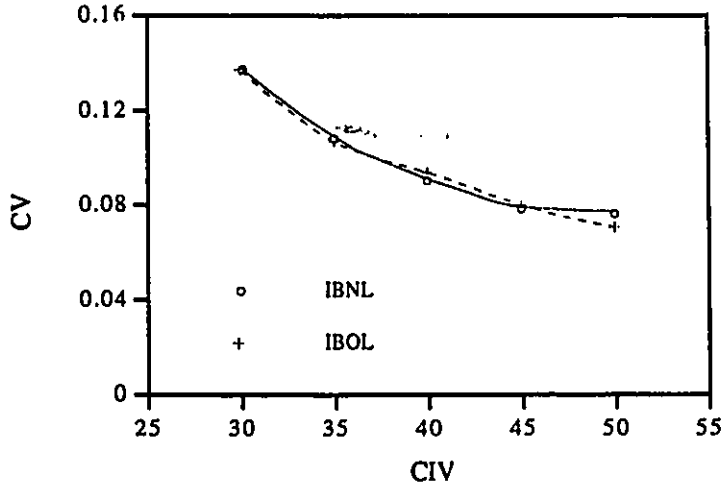


(b) Voice Packet Loss Probability

Fig. 5.16 Performance of RCP-II Using Finite Voice Buffer versus CIV under IBNL and IBOL



(c) Mean Data Message Delay



(d) Voice Delay Jitters (CV)

Fig. 5.16 Performance of RCP-II Using Finite Voice Buffer versus CIV under IBNL and IBOL

Fig. 5.16(a), (b), (c), and (d) show the influence of CIV on the mean voice packet delay, voice packet loss probability, mean data message delay, and voice delay jitters under IBNL and IBOL buffering methods, respectively. As before, for the buffering methods IBNL and IBOL, we still obtain almost the same voice packet delay, data message delay, and voice delay jitters, but the voice packet loss probability using IBOL is smaller than that using IBNL.

5.4. Discussion and Summary

When evaluating data traffic performance, we always fix those parameters related to voice load, such as N and M, $\overline{T}_{call}$ and $\overline{T}_{idle}$, and $\overline{T}_{talkspurt}$ and $\overline{T}_{silence}$, so that the total voice load is fixed, then enlarge data message arrival rate $\overline{R}_{message}$ to increase the overall network throughput. Sometimes we alter one of the voice parameters to change the total voice throughput and fix the total data load so that the overall network load is changed.

The results show that the data message delay almost totally depends on the overall network load. The data message delay is relatively small when the overall network load is very light. It increases as the overall network load increases and becomes very large when the overall network load is close to 1. Data message can function at reasonable performance when the overall network load is not heavy.

Voice performance depends very much on the protocol. If using RCP-I, mean voice packet delay grows when the overall network load increases, and it is finally capped at a maximum value as the total network load reaches a certain value. If using RCP-II, mean voice packet delay is kept at a certain value which is independent of the data traffic load. This is an important observation because it means real-time applications can be implemented so long as we keep the capped value below the maximum threshold tolerated by voice packets. Voice delay jitters depends on the protocol as well. Fig. 5.6 gives the comparison among RCP-I, RCP-II and FDDI. One will notice that RCP-II gives the smallest *CV*. The tradeoff is the larger mean voice packet delay. However, one point should be noted: real-time delivery does not mean that the smaller is the delay, the better is its performance. As long as RCP-II can satisfy the real-time requirement, voice packet synchronous output is preferred. Therefore we say that voice can obtain a much better performance by using RCP-II. But RCP-I is still kept because it is relatively simple in implementation.

We also evaluated the RCP-II performances when the network is used as a backbone network. It is obvious that the protocol works well, only with the difference that each station will transmit all voice packets waiting in the buffer pools. Four different buffering schemes for voice packets are discussed: infinite buffer, IBNL, SBNL, and IBOL. Although the performance results (including the mean voice packet delay, the voice delay jitters (CV), and the data message delay) from all those four buffering schemes are very close, they are slightly different. The shared buffer gives a lower loss probability but the

individual buffer gives a smaller voice delay jitters. Note that when the voice packet arrives and finds its buffer is full, if the first packet waiting in the buffer is lost rather than the newest arrival packet, a smaller loss probability and a shorter mean voice packet delay are achieved. This points out a new method for a small size of buffer pool: the packet already waiting for a long time should be lost instead of the new arrival packets, that is, the oldest packet (the longest waiting packet) should be lost. Of course, the smaller the buffer size is, the larger the voice packet loss probability will be. When looking at Fig. 5.11(a), with a buffer size of 2 using the IBOL method, the loss probability is around 1×10^{-3}.

One may question the maximum voice packet delay using RCP. If we could ensure that the token would be back to a certain station to serve voice packets in the exact time duration which is equal to the voice packet generation time $T_{interval}$ (that is, $L_{cycle} = T_{interval}$), and the supervisor station never changes, the maximum voice packet delay is equal to $T_{interval}$ and the delay jitters is zero. Unfortunately, the real situation is that $L_{cycle} > T_{interval}$ because of the ring latency $T_{latency}$, and the supervisor station keeps changing for reliability and antimonopoly reasons. Although we did not give the exact value of the maximum voice packet delay here, it should be less than twice the transmission cycle if all the waiting voice packets are transmitted during the token visiting this station. This condition is reflected in our protocol design, and is realistic since the RCP will be applied most probably to a backbone network.

One thing worth mentioning is that the simulation is very time consuming. For example, one set of curves (including voice packet delay, data message delay, and CV) needs to run for at least two days. It is obvious how much time is required to achieve the results. A better simulation tool is needed for the queueing network simulation.

Chapter 6

Design and Implementation Issues

6.1. Implementation

Through simulation, we obtained a large number of results under various situations which are represented in Chapter 5. To implement this protocol, a set of suitable parameters is needed in order to achieve the desirable performances. In this section, we will give the procedures on how to choose those parameters.

6.1.1. Determining CIV

All simulation results show that the parameter CIV determines the maximum mean voice packet delay. In order to find the relationship between them, Fig. 6.1. gives all the voice delay values versus CIV when the RCP-II is used, which are all obtained from Chapter 5.

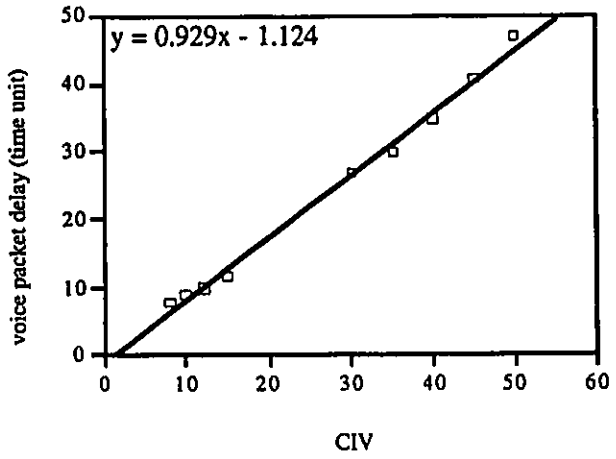


Fig. 6.1. Relationship between mean voice packet delay and CIV

Although those values are not under the same network conditions, we already

showed that voice packet delay has very little dependency on different voice load (Fig. 5.4(a) and Fig. 5.8(a)), different data load (Fig. 5.8(a)), and different number of stations on the network (Fig. 5.5(a)).

Fig. 6.1. presents a linear relationship between mean voice packet delay and CIV. From interpolation, we obtain an empirical formula which is $y = 0.929x - 1.124$, where y is the mean voice packet delay in time unit, and x is the value of CIV. Using this formula, we could say that it is safe to choose CIV as the required mean voice packet delay which is normalized by the packet transmission time P_{trans}. Because the transmission cycle length $L_{cycle} = T_{latency} + CIV \times P_{trans}$, and in general, ring latency is very small, we can say that we will choose the transmission cycle length as the required mean voice packet delay.

Obviously, we now want to determine what the packet transmission time P_{trans} should be.

As indicated in section 2.3, a continuous voice analog signal is digitized by a coder at each voice source. This digitizing rate is usually a constant, and is called a sample rate. A typical example is that the voice signal is sampled at 8 kHz and encoded into the 64 kbits/s PCM signal [WONG89]. After the sample rate is fixed, the total number of bits generated in one transmission cycle is given. If all those samples are packetized into one packet (this is an optimizing assumption from our design), and the network transmission rate is given, the packet transmission time P_{trans} is the packet length (including the overhead) in bits divided by the network transmission rate.

If L_{cycle} and P_{trans} are given, the CIV is determined.

6.1.2. Throughput and Performance

From equations in Section 4.1.1, voice and data traffic model, we know the following facts:

(1) the overall network throughput is the sum of the voice and the data throughput (from equation 6 in Chapter 4);

(2) the voice throughput depends on the total voice calls which are given by N and M (the number of stations on the network and the number of voice calls per station), the frequency that each voice call appears represented by $\overline{T}_{call}$ and $\overline{T}_{idle}$ (mean voice

call holding and idle periods, or called as off-hook and on-hook states), the talkspurt-silence characteristics of voice call indicated by $\overline{T}_{talkspurt}$ and $\overline{T}_{silence}$ (mean talkspurt and silence durations), and voice packet generating interval $T_{interval}$.

(3) data throughput depends on the data message arrival rate $\overline{R}_{message}$ (in number of messages per time unit) and the mean message length $\overline{L}_{message}$ (in number of packets per message).

Based on equations (1) to (6) in Chapter 4, we could calculate the throughput. For example, when only a single station attaches to each node (i.e., $M = 1$), if Table 1 parameters are used and $T_{interval} = 15$, the total voice throughput is

$$\frac{\overline{T}_{call}}{\overline{T}_{call}+\overline{T}_{idle}} \times \frac{\overline{T}_{talkspurt}}{T_{interval} \times (\overline{T}_{talkspurt}+\overline{T}_{silence})} \times M \times N$$
$$= \frac{4500}{4500+4500} \times \frac{225}{15 \times (225+180)} \times 1 \times 10$$
$$= 0.1852$$

total data throughput and the overall network throughput are shown in the following table:

voice throughput	0.1852									
$\overline{R}_{message}$	0.0004	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009
data throughput	0.04	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
total throughput	0.2252	0.2852	0.3852	0.4852	0.5852	0.6852	0.7852	0.8852	0.9852	1.0852

The second example is when this protocol is used on a backbone network ($M = 4$), and Table 1 parameters are used but $\overline{T}_{call} \to \infty$, $\overline{T}_{idle} = 0$, and $T_{interval} = 40$ (i.e., $CIV = 40$). The total voice throughput is

$$\frac{\overline{T}_{talkspurt}}{T_{interval} \times (\overline{T}_{talkspurt} + \overline{T}_{silence})} \times M \times N$$

$$= \frac{225}{40 \times (225 + 180)} \times 4 \times 10$$

$$= 0.5556$$

The data throughput and the overall network throughput are shown in the following table:

voice throughput	0.5556						
$\overline{R}_{message}$	0.0004	0.001	0.002	0.0024	0.003	0.0035	0.005
data throughput	0.04	0.1	0.2	0.24	0.3	0.35	0.5
total throughput	0.0.5956	0.6556	0.7556	0.7956	0.8556	0.9056	1.0556

As for any other protocol, the smaller the total throughput on the network is, the better the performance will be. The maximum total throughput on the network is close to 1.

6.1.3. Choosing Parameters

To summarize the above, we give the following as one possible way to design our network:

(1) The transmission cycle length L_{cycle} is the required mean voice packet delay.

(2) The voice packet length in bits is the overhead plus the product of the transmission cycle length L_{cycle} and the voice signal sample rate: overhead + L_{cycle} X sample rate.

(3) The data packet length is equal to the voice packet length.

(4) The packet transmission time P_{trans} is the packet length divided by the network transmission rate.

(5) The CIV (Counter Initial Value) is the transmission cycle length L_{cycle} divided by

the packet transmission time P_{trans} .

(6) For the buffering scheme, the oldest packet lost (rather than the newest arrival packet) is recommended because it achieves a smaller loss probability and a shorter mean voice packet delay. The shared or individual buffer should be chosen according to the requirements. The shared buffer gives a lower loss probability but the individual buffer gives a smaller voice delay jitters. Of course, the smaller the buffer size is, the larger the voice packet loss probability will be.

(7) The network performance could be determined if the voice and data traffic throughput is given, as shown in section 6.1.2.

6.2. Fault Recovery Procedure

Reliability addresses the issue of a network's ability to continue operations in the face of failures. A reliable network is one that can continue operations despite the failure of some critical element. There are several kinds of fault that may happen to the ring networks: link failure, station failure, and token failure. A single failure will disrupt the operation of the whole network. Therefore, some techniques have to be used to overcome this deficiency [BUX89, STAL90, TANE88].

The fault recovery is a very big issue, and there are numerous failure situations. The above discussion could only cover some very basic and typical scenarios.

6.2.1. Recovery of Link Failure

One of the recovery techniques is to use one or more wiring concentrator (or wire center in [TANE88]). While logically still a ring, physically each station is connected to the wire concentrator. Wiring concentrators have the ability to detect and bypass faults that occur in the ring segments between concentrators. Link faults are bypassed by the associated relays in the wiring concentrator.

As shown in Fig. 6.2, inside the wire concentrator are bypass relays that are energized by current from the stations. If the ring breaks or a station goes down, loss of the drive current will release the relay and bypass the station. The relay can also be operated by

software to permit diagnostic programs to remove stations one at a time to find faulty stations and ring segments. The ring can then continue operation with the bad segment bypassed. When a network consists of many clusters of stations far apart, a topology with multiple wire centers can be used. Let us suppose that the cable to one of the stations in Fig. 6.2 was replaced by a cable to a distant wire center.

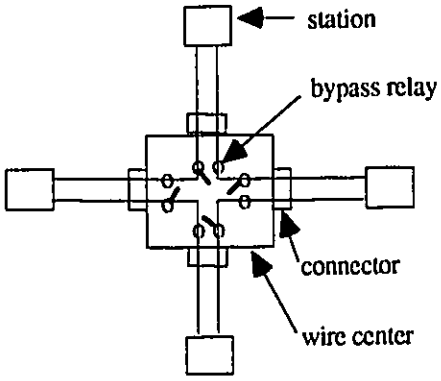
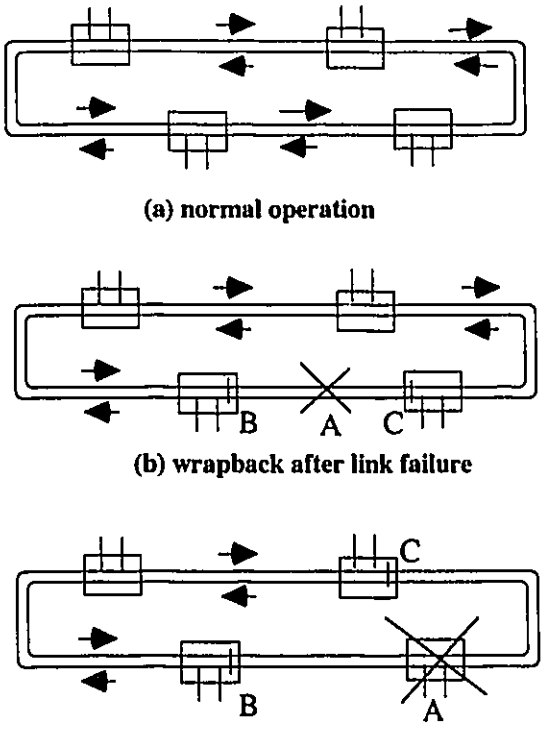


Fig. 6.2. Four Stations Connected via a wire center

Another approach that enhances the ring's reliability is by using a dual-ring structure as in FDDI. The network consists of two rings, one transmitting clockwise and the other transmitting counterclockwise. If either one breaks, the other can be used as a backup. If both break at the same point (for example, point A in Fig. 6.3(b)), due to a fire or other accident, the wiring concentrator on either side of the broken link "wrap back" (that is, the point B and C in Fig. 6.3(b)), so that the two rings can be joined into a single ring approximately twice as long, i.e., reconfigure internally in such a way that the failed link is eliminated (see Fig. 6.3(a) for normal operation and Fig. 6.3(b) for wrapback after link failure). When the fault is covered, the network will reconfigure itself.



(a) normal operation

(b) wrapback after link failure

(c) wrapback after concentrator failure

Fig. 6.3. Dual Ring Configuration

6.2.2. Recovery of Station Failure

A single user station failure on a ring network can be easily bypassed by use of the wiring concentrator, and this station is removed from the network, as shown in Fig. 6.2. If a wiring concentrator fails, a reconfiguration can be performed by dual ring configuration as shown in Fig. 6.3(c). The whole ring can still run normally. The bandwidth it was occupied is automatically released and the other voice or data packet can use it.

The supervisor station performs the monitoring function of overseeing the ring, and every station has the capability of becoming the supervisor. If the supervisor goes down when the frame control bit C=1, the other station will become a new supervisor. But if the supervisor goes down when the frame control bit C=0, the other stations will find that they are only allowed to transmit voice packets. Of course, this may be caused by the heavily loaded voice traffic. Therefore, if a station has too many data packets waiting, it should send a QUERY control frame to examine the supervisor station. If it does not get a

response, it can transmit a CLAIM SUPERVISOR control frame. If this frame circumnavigates the ring before any other CLAIM SUPERVISOR control frames, the sender becomes the new supervisor station.

6.2.3. Recovery of Token Failure

Another serious fault of token passing networks is the possibility of damage to, or loss of, the token. The supervisor station is responsible for the damaged token by comparing the received token with the normal token format. To detect a lost-token condition, the supervisor has a timer that is set to the longest possible tokenless interval, namely, each station transmitting for the full token-holding time. The timer is reset after every valid token or frame. If this timer goes off, the supervisor drains the ring and issues a new token.

Chapter 7

Conclusion

We have proposed a token-passing protocol for integrated services. The major feature is the inclusion of a rotation counter in the token. Two variations of the protocol are proposed: the RCP-I and RCP-II, each depending on the algorithm taken by a supervisor station. Token format design and details of the operation have been given.

The supervisor station can control the performance of each station by setting the values to various control fields to meet different delay performance demands. This is mainly achieved by using the counter that limits the amount of traffic allowed into the network in each cycle, thus allowing the performance of all stations to be correlated.

The modeling and simulation of the system are performed to evaluate our RCP performance for a voice and data integration application. The influence of various network conditions is studied. These include the service disciplines, the network size, the single/multiple voice arrivals per station, the varied voice/data load, the CIV, the symmetric/asymmetric environment, the infinite/finite buffer, the shared/individual buffer, the newest/oldest packet lost, and so on. The measures of performance are the mean voice packet delay, the voice delay jitters (CV), the voice packet loss probability, the mean data message delay, and the normalized network throughput.

The comparison among the RCP-I, RCP-II, and the FDDI is executed as well. The results demonstrate that the protocol is adequate for integrated services. The RCP-I can support integrated services with a performance comparable to the FDDI protocol. RCP-II can be desirable for real-time synchronous data such as voice because of its better ability to

meet the basic requirements, i.e., a more constant mean voice packet delay and a smaller voice delay jitters with respect to network load. The recommended procedure to choose network parameters is given.

We have compared different buffering schemes: infinite buffer, IBNL (Individual Buffer and the Newest Lost), IBOL (Individual Buffer and the Oldest Lost), and SBNL (Shared Buffer and the Newest Lost). We have found that it is advantageous to push out the oldest packet (the first packet waiting inside the buffer) instead of the newest arrival packet, if the buffer size is finite so that the voice packet loss exists. This will give a smaller voice packet loss probability and a shorter mean voice packet delay.

In summary, the RCP is a potential candidate for use with high-speed token rings.

References

ANSI88 American National Standards Institute, *FDDI Token Ring Media Access Control,* ANSI Standard X3T9.5, 1988.

BRAD59 P. T. Brady, "A statistical analysis of On-Off patterns in 16 conversations," Bell Systems Technical Journal, pp. 353-364, Mar. 1959.

BUX83 W. Bux and M. Schlatter, "An approximation method for the performance analysis of buffer insertion rings," *IEEE Trans. Commun.*, vol. COM-31, pp. 50-55, Jan. 1983.

BUX85 W. Bux and D. Grillo, "Flow Control in local-area networks of interconnected token rings," *IEEE Trans. commun.*, vol. COM-33, no. 10, pp. 1058-1065, Oct. 1985.

BUX89 W. Bux, "Token-Ring Local-Area Networks and Their Performance", *Proc. IEEE,* vol. 77, no. 2, pp. 238-256, Feb. 1989.

CHEN93 W.-T. Chen and J.-H. Yu, "Distributed protocol for integrated voice/data token passing ring networks," *Comput. Commun.*, vol. 6, no. 6, pp. 338-349, June 1993.

CHOI90 M. Choi and C. M. Krishna, "An Adaptive Algorithm to Ensure Differential Service in a Token-Ring Network," *IEEE Trans. Comput.*, vol. 39, no. 1, pp. 19-33, Jan. 1990.

DePR91 M. De Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Ellis Horwood, 1991.

FERG85 M. J. Ferguson and Y. J. Aminetzah, "Exact Results for Asymmetric Token Ring Systems," *IEEE Trans. Commun.*, vol. COM-33, no. 3, pp. 223-231, Mar. 1985.

FORT89 P. J. Fortier, *Handbook of LAN technology*, Intertext, 1989.

FUHR88 S. W. Fuhrmann and Y. T. Wang, "Analysis of Cyclic Service Systems with Limited Service: Bounds and Approximations," *Performance Evaluation*, vol. 9, no. 1, pp. 35-54, 1988.

GARZ90 R. F. Garzia and M. R. Garzia, "Introduction to modeling and simulation," *Network Modeling, Simulation, and Analysis*, R. F. Garzia and M. R. Garzia (ed.), pp. 1-20, Marcel Dekker Inc., 1990.

GROW82 R. M. Grow, "A time token protocol for local area networks," *Electro'82, Token Access Protocols*, paper 17/3, pp. 1-7, 1982.

GRUB83 J. G. Gruber and N. H. Le, "Performance requirements for integrated voice/data networks," *IEEE J. Select. Areas Commun.*, vol. SAC-1, no. 6, pp. 981-1005, Dec. 1989.

HASH72 O. Hashida, "Analysis of Multiqueue," *Rev. Elec. Commun. Lab.*, vol. 20, pp. 189-199, Mar.-Apr., 1972.

HILA92 W. Hilal and M. T. Liu, "Register-insertion: a protocol for the next generation of ring local-area networks," *Computer Networks and ISDN Systems*, vol. 24, no. 5, pp. 349-366, June 1992.

HOPP88 A. Hopper, R. M. Needham, "The Cambridge fast ring networking system," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1214-1223, Oct. 1988.

IEEE85a IEEE Computer-Society, *Token ring access method and physical layer specifications,* ANSI/IEEE Standard 802.5-1985 .

IEEE85b IEEE Computer-Society, *Logic Link Control,* ANSI/IEEE Standard 802.2-1985 .

KARV86 D. Karvelas and A. Leon-Garcia, "Performance of Integrated Packet Voice/Data Token-Passing Rings", *IEEE J. Select Areas Commun.*, vol. SAC-4, no. 6, pp. 823-832, Sept. 1986.

KLEI75 L. Kleinrock, *Queueing Systems*, vol. 1. Theory, John Wiley & Sons, 1975.

KUEH79 P. J. Kuehn, "Multiqueue System with Nonexhaustive Cyclic Service," *Bell*

Syst. Tech. J., vol. 58, pp. 671-698, Mar. 1979.

MARK90 J. W. Mark and B. J. Lee, "A dual-ring LAN for integrated voice/video/data services," *Proc. IEEE INFOCOM'90*, pp. 850-857, 1990

OKUD92 T. Okuda, H. Akimaru, and K. Nagai, "Performance evaluation for multiclass traffic in ATM systems," *Proc. IEEE ICC'92*, pp. 207-211, 1992.

QNAP84 *QNAP2 reference manual,* Bull and INRIA, 1983, 1984.

ROSS87 F. E. Ross, "Rings are 'round for good!" *IEEE Network Mag.*, pp. 31-38, Jan. 1987.

ROSS89 F. E. Ross, "An overview of FDDI: the fiber distributed data interface," *IEEE J. Select. Areas Commun.*, vol. 7, no. 7, pp. 1043-1051, Sept. 1989.

SARK89 D. Sarkar and W. I. Zangwill, "Expected Waiting Time for Nonsymmetric Cyclic Queueing Systems — Exact Results and Applications", *Management Science*, vol. 35, no. 12, Dec. 1989.

SEVC87 K. C. Sevcik and M. J. Johnson, "Cycle Time Properties of the FDDI Token Ring Protocol," *IEEE Trans. Software Eng.*, vol. SE-13, no. 3, pp. 376-385, Mar. 1987.

SONG93a J. Song and O. Yang, "An Integrated Services Protocol for Token Rings", submitted to *IEEE Trans. Networking*, 1993.

SONG93b J. Song and O. Yang, "Rotation Counter Protocol – A Token-Ring Protocol for Integrated Services", *IEEE Proc. 18th Conf. on Local Computer Networks*, Minneapolis, Minnesota USA, pp. 474-481, 1993.

SPIE91 E. M. Spiegel, "Performance Analysis of the Timed-Token Protocol: A Vacation Model," *High Speed Networking, III*, O. Spaniol and A. Danthine (eds.), pp. 109-124, 1991.

STAL90 W. Stallings, *Handbook of Computer Communications Standards, Volume 2: Local Area Network Standards*, Howard W. Sams & Company, 1990.

STAL93 W. Stallings, *Local and Metropolitan Area Networks*, Macmillan, 1993.

SUDA89 T. Suda and T. T. Bradley, "Packetized voice/data integrated transmission on a token passing ring local area network," *IEEE Trans. commun.*, vol. 37, no. 3, pp. 238-244, Mar. 1989.

TAKA90 H. Takagi, "Queueing Analysis of Polling Models: An Update", *Stochastic Analysis of Computer and Communication Systems*, H. Takagi (ed.), pp. 267-318, 1990.

ULM82 J. M. Ulm, "A timed token ring local area network and its performance characteristics," *IEEE Proc. 7th Conf. on Local Computer Networks,* pp. 50-60, 1982.

WALR91 J. Walrand, *Communication Networks: A First Course,* Aksen Associates, 1991.

WILK79 M. V. Wilkes and D. J. Wheeler, "The Cambridge digital communication ring," *Proc. Local Area Commun. Networks Symp.*, pp. 47-61, May 1979.

WILL92 M. Willebeek-LeMair, F. Schaffa, and B. Patel, "Isochronous versus synchronous traffic in FDDI," *IEEE Proc. 17th Conf. on Local Computer Networks*, pp. 100-109, 1992.

WONG84 J. W. Wong and P. M. Gopal, "Analysis of a token ring protocol for voice transmission", *Computer Networks*, vol. 8, no. 4, pp. 339-346, 1984.

WONG89 P.-C. Wong and T. P. Yum, "An integrated services token-controlled ring network," *IEEE J. Select. Areas Commun.*, vol. 7, no. 5, pp. 670-679, June 1989.

YANG92 Q. Yang, D. Ghosal, and S. K. Tripathi, "Performance study of two protocols for voice/data integration on ring networks," *Computer Networks and ISDN Systems*, vol. 23, no. 4, pp. 267-285, Jan. 1992.

Appendix

QNAP2 Simulation Codes

A. Simulation Code for the Timed-Token Protocol in FDDI

```
/DECLARE/
   INTEGER
       N=10,           & number of stations on the network
       pnb(N),         & number of voice packets in the talkspurt
       dpnb,           & number of data packets in the message
       dataloss,       & number of data packets lost
       i, j;

   REAL
       mleng,          & mean data message length
       pleng,          & data packet length
       ptran,          & one packet transmission time
       walk,           & walktime between two stations
       mholding,       & mean voice call holding time
       midle,          & mean voice call idle time
       mlasting,       & mean talkspurt lasting time
       msilent,        & mean silence period duration
       vpgt,           & voice packet generation time
       holding(N),     & voice call holding time
       last(N),        & talkspurt lasting time
       tsilent(N),     & silence lasting time
       TTRT=3.0,       & Target Token Rotation Time
       time1(N),       & token last visiting time
       time2(N),       & token present visiting time
       TRT(N),         & Token Rotation Timer
       THT(N),         & Token Holding Timer
       mrtv,           & mean voice packet delay
       mrtd,           & mean data message delay
       civ,            & confidence interval of mrtv
       cid,            & confidence interval of mrtd
       variance,       & variance of mrtv
       dvarian,        & variance of mrtd
       midt,           & mean voice packet interdeparture time at destination
       vidt,           & variance of midt
       ciidt,          & confidence interval of midt
       x, y;

   QUEUE INTEGER
       id,             & index of station number
```

```
        hook,           & indicating status of on-hook(0)/off-hook(1)
        silent;         & indicating talkspurt(1)/silence(0)

    QUEUE REAL
        t;              & mean data message interarrival time

    QUEUE
        s(N),           & data stations
        pkt(N),         & packetization for data messages
        source(N),      & data messages arrival
        server(N),      & serving voice packets
        vs(N),          & voice stations
        vpkt(N),        & packetization for talkspurts
        vcall(N),       & generating voice calls
        talkspur(N),    & generating talkspurts
        vpid(N),        & voice packet interdeparture after transmission
        time(N),        & measuring data message response time
        token,          & token
        init;           & initial queue

    FLAG
        vtrans(N),      & control voice packet transmission
        transmit(N),    & control data packet transmission
        circ,           & control token circulation
        endcall(N),     & indicating voice call end
        interd(N);      & indicating voice packet departure

    CUSTOMER INTEGER
        SEQ;

    REF CUSTOMER
        C,
        cv;


/CONTROL/ OPTION = NRESULT, NSOURCE;
         MARGINAL = time, vpid, vs;


&-------------------------------------------------------------
& This is for generating voice call.
&    voice call will last for sometime (mholding), and then this
&    voice station will be idle for sometime (midle).
&    Two states (holding and idle) will be alternative.
&-------------------------------------------------------------
/STATION/ NAME = vcall;
          TYPE = SOURCE;
       SERVICE = BEGIN
                 IF vcall(id).hook=1 THEN
                    BEGIN
                    holding(id) := EXP(mholding);
                    CST(holding(id));
                    vcall(id).hook := 0;
                    END
                 ELSE
                    BEGIN
                    RESET(endcall(id));
                    EXP(midle);
```

```
                    SET (endcall(id));
                    vcall(id).hook := 1;
                    END;
                  TRANSIT(OUT);
                  END;


&-----------------------------------------------------------------
& During each voice call holding time, there will be talkspurt
& and silence alternatively. Talkspurt will last for EXP(mlasting),
& then the following will be silent for EXP(msilent).
&-----------------------------------------------------------------
/STATION/ NAME = talkspur;
          TYPE = SOURCE;
       SERVICE = BEGIN
                 IF vcall(id).hook = 1 THEN
                   BEGIN
                   IF talkspur(id).silent = 1 THEN
                     BEGIN
                     last(id) := EXP(mlasting);
                     IF last(id) < holding(id) THEN
                       holding(id) := holding(id)-last(id)
                     ELSE
                       last(id) := holding(id);
                     TRANSIT(NEW(CUSTOMER),vpkt(id));
                     CST(last(id));
                     talkspur(id).silent := 0;
                     END
                   ELSE
                     BEGIN
                     tsilent(id) := EXP(msilent);
                     IF tsilent(id) < holding(id) THEN
                       holding(id) := holding(id)-tsilent(id)
                     ELSE
                       tsilent(id) := holding(id);
                     CST(tsilent(id));
                     talkspur(id).silent := 1;
                     END;
                   END
                 ELSE
                   BEGIN
                   talkspur(id).silent := 1;
                   WAIT(endcall(id));
                   END;
                 TRANSIT(OUT);
                 END;


&-------------------------
& generating voice packets
&-------------------------
/STATION/ NAME = vpkt;
       SERVICE = BEGIN
                 y := last(id);
                 pnb(id) := INTREAL(y/vpgt);
                 IF pnb(id) > 0 THEN
                   FOR i := 1 STEP 1 UNTIL pnb(id) DO
                     BEGIN
```

```
                    CST(vpgt);
                    cv :=  NEW(CUSTOMER);
                    cv.SEQ :=0;
                  TRANSIT(cv, vs(id));
                  END;
                SEQ := 2;
                TRANSIT(vs(id));
                END;


&--------------
& voice station
&--------------
/STATION/ NAME = vs;
       SERVICE = BEGIN
                 WAIT(vtrans(id));
                 RESET(vtrans(id));
                 TRANSIT(server(id));
                 END;


&---------------------------
& voice packets transmitting
&---------------------------
/STATION/ NAME = server;
       SERVICE = BEGIN
                 CST(ptran);
                 THT(id) := THT(id) - ptran;
                 IF (THT(id) > 0) AND
                    (vs(id).NB >1) THEN
                   BEGIN
                   SET(vtrans(id));
                   END
                 ELSE
                   BEGIN
                   SET(circ);
                   END;
                 SET(interd(id));
                 RESET(interd(id));
                 IF SEQ = 2 THEN
                   BEGIN
                   TRANSIT(OUT);
                   END;
                 TRANSIT(vpid(id));
                 END;


&---------------------------------
&  voice packet interdeparture time
&---------------------------------
/STATION/ NAME = vpid;
       SERVICE = BEGIN
                 WAIT(interd(id));
                 TRANSIT(OUT);
                 END;
```

```
&-----------------------------------------------
& It's for measuring the message response time
&    - from generation to finishing transmission
&    - by P/V procedure
&-----------------------------------------------
/STATION/ NAME = time(1 STEP 1 UNTIL N);
          TYPE = RESOURCE, INFINITE;


&--------------------
& generating messages
&--------------------
/STATION/ NAME = source;
          TYPE = SOURCE;
       SERVICE = BEGIN
                 EXP(1./t);
                 P(time(id));
                 TRANSIT(pkt(id));
                 END;


&--------------
& packetization
&--------------
/STATION/ NAME = pkt;
       SERVICE = BEGIN
                 x := EXP(mleng);
                 dpnb :=  INTREAL(x/pleng);
                 IF dpnb > 0 THEN
                   FOR i := 1 STEP 1 UNTIL dpnb DO
                     BEGIN
                     C := NEW(CUSTOMER);
                     C.SEQ :=  0;
                     IF CUSTNB(s(id)) >= 500 THEN
                       BEGIN
                       dataloss := dataloss + 1;
                       TRANSIT(C, OUT);
                       END
                     ELSE TRANSIT (C, s(id));
                     END;
                 SEQ := 1;
                 TRANSIT(s(id));
                 END;


&-------------
& user station
&-------------
/STATION/ NAME = s;
       SERVICE = BEGIN
                 WAIT(transmit(id));
                 RESET(transmit(id));
                 CST(ptran);
                 THT(id) := THT(id) - ptran;
                 IF (THT(id) > 0) AND
                    (s(id).NB >1) THEN
                   BEGIN
                   SET(transmit(id));
```

```
                  END
                ELSE
                  BEGIN
                  SET(circ);
                  END;
                IF SEQ = 1 THEN V(time(id));
                TRANSIT(OUT);
                END;


&-------------------------
& initial queue
&   --- to set circ = true
&-------------------------
/STATION/ NAME = init;
          INIT = 1;
       SERVICE = SET (circ);
       TRANSIT = OUT;


&-------------------------------------------------------------
& token circulating in ring
&   It is based on FDDI MAC protocol. See Chapter 3 for details
&-------------------------------------------------------------
/STATION/ NAME = token;
          INIT = 1;
       SERVICE = BEGIN
                 WAIT(circ);
                 CST(walk);
                 time1(j) := time2(j);
                 time2(j) := TIME;
                 TRT(j) := time2(j)-time1(j);
                 THT(j) := TTRT - TRT (j);
                 IF THT(j) < 0 THEN
                 &--means token came back late
                   BEGIN
                   time2(j) := time2(j) + THT(j);
                   END;
                 IF vs(j).NB > 0 THEN
                 &--voice packet are waiting ==> transmit.
                   BEGIN
                   SET(vtrans(j));
                   RESET(circ);
                   END;
                 WAIT(circ);
                 IF (THT(j) > 0) AND (s(j).NB >0) THEN
                 &--transmit data packets if there is time left
                   BEGIN
                   SET(transmit(j));
                   RESET(circ);
                   END;
                 j := j +1;
                 IF j > N THEN j :=1;
                 TRANSIT(token);
                 END;
```

```
&------------------
& TMAX and printout
&------------------
/CONTROL/ TMAX = 150000;
      ACCURACY = vtime, time, vpid, vs;
         ENTRY = BEGIN
                 PRINT(" ");
                 PRINT(" ");
                 PRINT("input parameters:");
                 PRINT(" number of stations", N);
                 PRINT(" voice call holding time", mholding);
                 PRINT(" voice station idle time", midle);
                 PRINT(" talkspurt last time", mlasting);
                 PRINT(" silent time", msilent);
                 PRINT(" voice packet generating time", vpgt);
                 PRINT(" data arrival rate(message/ms)", source(1).t);
                 PRINT(" message length", mleng);
                 PRINT(" packet length", pleng);
                 PRINT(" walk time between two stations", walk);
                 PRINT(" TTRT", TTRT);
                 PRINT(" ");
                 END;
         EXIT = BEGIN
           PRINT("results");
           PRINT(" mean data response for 1", MRESPONSE(time(1)));
           PRINT(" mean data response for 4", MRESPONSE(time(4)));
           PRINT(" mean data response for 7", MRESPONSE(time(7)));
           PRINT(" mean data response for 10",MRESPONSE(time(10)));
           PRINT(" mean voice response for 1", MRESPONSE(vs(1)));
           PRINT(" mean voice response for 4", MRESPONSE(vs(4)));
           PRINT(" mean voice response for 7", MRESPONSE(vs(7)));
           PRINT(" mean voice response for 10",MRESPONSE(vs(10)));
           PRINT("mean v.p. interdeparture for 1", MRESPONSE(vpid(1)));
           PRINT("  variance of interdeparture", VRESPONSE(vpid(1)));
           PRINT("mean v.p. interdeparture for 4", MRESPONSE(vpid(4)));
           PRINT("  variance of interdeparture", VRESPONSE(vpid(4)));
           PRINT("mean v.p. interdeparture for 7", MRESPONSE(vpid(7)));
           PRINT("  variance of interdeparture", VRESPONSE(vpid(7)));
           PRINT("mean v.p. interdeparture for 10", MRESPONSE(vpid(10)));
           PRINT("  variance of interdeparture", VRESPONSE(vpid(10)));
                 END;


&-----------------------------------------
& initialization, simulation, and calculation
&-----------------------------------------
$MACRO exec(arr, leng)
/EXEC/ BEGIN
       FOR i := 1 STEP 1 UNTIL N DO
         BEGIN
         vpid(i).id := i;
         server(i).id := i;
         vs(i).id := i;
         vpkt(i).id := i;
         vcall(i).id := i;
         talkspur(i).id := i;
         s(i).id := i;
         pkt(i).id := i;
```

```
        source(i).id := i;
        source(i).t := arr;
        vcall(i).hook := 0;
        talkspur(i).silent := 1;
        END;
      j := 1;
      mholding := 900.0;
      midle := 900.0;
      mlasting := 45.0;
      msilent := 36.0;
      vpgt := 3;
      mleng :=leng;
      pleng := 1;
      ptran := 0.2;
      walk := 0.001;
      time1 := 0;
      time2 := 0;
      THT := 0;
      mrtv := 0;
      civ := 0;
      mrtd := 0;
      cid := 0;
      variance := 0;
      dvarian := 0;
      midt := 0;
      vidt := 0;
      ciidt := 0;
      dataloss := 0;
      SIMUL;
      FOR i := 1 STEP 1 UNTIL N DO
        BEGIN
        mrtv := mrtv + (MRESPONSE(vs(i))+ptran)/N;
        civ := civ + CRESPONSE(vs(i))/N;
        mrtd := mrtd + MRESPONSE(time(i))/N;
        cid := cid + CRESPONSE(time(i))/N;
        variance := variance + VRESPONSE(vs(i))/N;
        dvarian := dvarian + VRESPONSE(time(i))/N;
        midt := midt + MRESPONSE(vpid(i))/N;
        vidt := vidt + VRESPONSE(vpid(i))/N;
        ciidt := ciidt + CRESPONSE(vpid(i))/N;
        END;
      PRINT(" mean response for voice", mrtv);
      PRINT(" variance of voice delay", variance);
      PRINT(" confidence interval for mrtv", 100*civ/mrtv, "%");
      PRINT(" mean response for data", mrtd);
      PRINT(" variance of data delay", dvarian);
      PRINT(" confidence interval for mrtd", 100*cid/mrtd, "%");
      PRINT(" mean interdeparture time for voice", midt);
      PRINT(" variance of interdeparture time for voice", vidt);
      PRINT(" confidence interval for midt", 100*ciidt/midt, "%");
      PRINT(" data packet loss", dataloss);
      END;
$END


&------------------
& change parameters
&------------------
```

```
$exec(0.002, 10)
$exec(0.010, 10)
$exec(0.020, 10)
$exec(0.030, 10)
$exec(0.035, 10)
$exec(0.038, 10)
$exec(0.040, 10)
$exec(0.045, 10)


/END/
```

B. Simulation Code for the RCP-I

```
/CONTROL/
    OPTION    =  NSOURCE;

/DECLARE/
    INTEGER
       N=10,           & number of stations on the network
       priority,       & the token is for voice (1) or data (2)
       super,          & the supervisor station
       dataloss,       & data packets lost when the network is saturated
       CIV=10,         & Counter Initial Value
       counter,        & counter for number of rotation rounds
       late,           & the L bit in the token
       pnb(N),         & number of voice packets in a talkspurt
       dpnb,           & number of data packets in a message
       i, j;

    REAL
       mleng,          & mean data message length
       pleng,          & data packet length
       ptran,          & one packet transmission time
       walk,           & walktime between two stations
       mholding,       & mean voice call holding time
       midle,          & mean voice call idle time
       mlasting,       & mean talkspurt lasting time
       msilent,        & mean silence period duration
       vpgt,           & voice packet generation time
       holding(N),     & a voice call holding time
       last(N),        & a talkspurt lasting time
       tsilent(N),     & a silence lasting time
       mrtv,           & mean voice packet delay
       mrtd,           & mean data message delay
       civ,            & confidence interval of mrtv
       cid,            & confidence interval of mrtd
       variance,       & variance of mrtv
       dvarian,        & variance of mrtd
       midt,           & mean voice packet interdeparture time at destination
       vidt,           & variance of midt
       ciidt,          & confidence interval of midt
       x, y;

    QUEUE INTEGER
       id,             & index of station number
       hook,           & indicating status of on-hook(0)/off-hook(1)
       silent;         & indicating talkspurt(0)/silence(1)

    QUEUE REAL
       t;              & mean data message interarrival time

    QUEUE
       s(N),           & data stations
       pkt(N),         & packetization for data messages
```

```
      source(N),      & data messages arrival
      server(N),      & serving voice packets
      vs(N),          & voice stations
      vpkt(N),        & packetization for talkspurts
      vcall(N),       & generating voice calls
      talkspur(N),    & generating talkspurts
      vpid(N),        & voice packet interdeparture after transmission
      time(N),        & measuring data message response time
      token,          & token
      init;           & initial queue

   FLAG
      vtrans(N),      & control voice packet transmission
      transmit(N),    & control data packet transmission
      circ,           & control token circulation
      endcall(N);     & indicating voice packet departure

   CUSTOMER INTEGER
      SEQ;

   REF CUSTOMER
      C;


/CONTROL/
   OPTION   =  NRESULT;
   MARGINAL =  time, vpid, vs;


&-----------------------------------------------------------------
& This is for generating voice call.
&    voice call will last for sometime (mholding), and then this
&    voice station will be idle for sometime (midle).
&    Two states (holding and idle) will be alternative.
&-----------------------------------------------------------------
/STATION/
   NAME     =  vcall;
   TYPE     =  SOURCE;
   SERVICE  =  BEGIN
               IF vcall(id).hook=1 THEN
                  BEGIN
                  holding(id) := EXP(mholding);
                  CST(holding(id));
                  vcall(id).hook := 0;
                  END
               ELSE
                  BEGIN
                  RESET(endcall(id));
                  EXP(midle);
                  SET (endcall(id));
                  vcall(id).hook := 1;
                  END;
               TRANSIT(OUT);
               END;
```

```
&-------------------------------------------------------------------
& During each voice call holding time, there will be talkspurt
& and silence alternatively. Talkspurt will last for EXP(mlasting),
& then the following will be silent for EXP(msilent).
&-------------------------------------------------------------------
/STATION/
     NAME      =  talkspur;
     TYPE      =  SOURCE;
     SERVICE   =  BEGIN
                  IF vcall(id).hook = 1 THEN
                     BEGIN
                     IF talkspur(id).silent = 1 THEN
                        BEGIN
                        last(id) := EXP(mlasting);
                        IF last(id) < holding(id) THEN
                           holding(id) := holding(id)-last(id)
                        ELSE
                           last(id) := holding(id);
                        TRANSIT(NEW(CUSTOMER),vpkt(id));
                        CST(last(id));
                        talkspur(id).silent := 0;
                        END
                     ELSE
                        BEGIN
                        tsilent(id) := EXP(msilent);
                        IF tsilent(id) < holding(id) THEN
                           holding(id) := holding(id)-tsilent(id)
                        ELSE
                           tsilent(id) := holding(id);
                        CST(tsilent(id));
                        talkspur(id).silent := 1;
                        END;
                     END
                  ELSE
                     BEGIN
                     talkspur(id).silent := 1;
                     WAIT(endcall(id));
                     END;
                  TRANSIT(OUT);
                  END;


&-------------------------
& generating voice packets
&-------------------------
/STATION/
     NAME      =  vpkt;
     SERVICE   =  BEGIN
                  y := last(id);
                  pnb(id) := 1 + INTREAL(y/vpgt);
                  FOR i := 1 STEP 1 UNTIL pnb(id) DO
                     BEGIN
                     CST(vpgt);
                     TRANSIT(NEW(CUSTOMER), vs(id));
                     END;
                  TRANSIT(OUT);
                  END;
```

```
&--------------
& voice station
&--------------
/STATION/
     NAME      =  vs;
     SERVICE   =  BEGIN
                  WAIT(vtrans(id));
                  RESET(vtrans(id));
                  TRANSIT(server(id));
                  END;


&---------------------------
& voice packets transmitting
&---------------------------
/STATION/
     NAME      =  server;
     SERVICE   =  BEGIN
                  CST(ptran);
                  counter := counter - 1;
                  SET(circ);
                  SET(interd(id));
                  RESET(interd(id));
                  IF SEQ = 2 THEN
                     BEGIN
                     TRANSIT(OUT);
                     END;
                  TRANSIT(vpid(id));
                  END;


&---------------------------------
&  voice packet interdeparture time
&---------------------------------
/STATION/
   NAME        =  vpid;
   SERVICE     =  BEGIN
                  WAIT(interd(id));
                  TRANSIT(OUT);
                  END;


&-----------------------------------------------
& It's for measuring the message response time
&    - from generation to finishing transmission
&    - by P/V procedure
&-----------------------------------------------
/STATION/
     NAME      =  time(1 STEP 1 UNTIL N);
     TYPE      =  RESOURCE, INFINITE;


&------------------
& generating message
&------------------
/STATION/
     NAME      =  source;
     TYPE      = SOURCE;
```

```
     SERVICE   =  BEGIN
                  EXP(1./t);
                  P(time(id));
                  TRANSIT(pkt(id));
                  END;


&--------------
& packetization
&--------------
/STATION/
     NAME      =  pkt;
     SERVICE   =  BEGIN
                  IF CUSTNB(s(id)) > 1000 THEN
                     TRANSIT(OUT);
                  x := EXP(mleng);
                  dpnb := INTREAL(x / pleng);
                  IF dpnb > 0 THEN
                  FOR i := 1 STEP 1 UNTIL dpnb DO
                     BEGIN
                     C := NEW(CUSTOMER);
                     C.SEQ := 0;
                     TRANSIT (C, s(id));
                     END;
                  SEQ := 1;
                  TRANSIT(s(id));
                  END;

&-------------
& user station
&-------------
/STATION/
     NAME      =  s;
     SERVICE   =  BEGIN
                  WAIT(transmit(id));
                  RESET(transmit(id));
                  CST(ptran);
                  counter := counter - 1;
                  IF (counter > 0) AND (s(id).NB >1) THEN
                     BEGIN
                     SET(transmit(id));
                     END
                  ELSE
                     BEGIN
                     IF counter = 0 THEN
                        BEGIN
                        counter := CIV;
                        priority := 1;
                        IF id < N THEN
                           super := id +1
                        ELSE
                           super := 1;
                        END;
                     SET(circ);
                     END;
                  IF SEQ = 1 THEN V(time(id));
                  TRANSIT(OUT);
                  END;
```

```
&-------------------------
& initial queue
&   --- to set circ = true
&-------------------------
/STATION/
     NAME     =  init;
     INIT     =  1;
     SERVICE  =  SET (circ);
     TRANSIT  =  OUT;


&-----------------------------------------------------------------
& token circulating in ring
&   This is based on RCP-I MAC protocol. See Chapter 3 for details
&-----------------------------------------------------------------
/STATION/
     NAME     =  token;
     INIT     =  1;
     SERVICE  =  BEGIN
                 WAIT(circ);
                 CST(walk);
                 IF priority = 1 THEN
                 &--token serves voice packets
                    BEGIN
                    IF vs(j).NB >0 THEN
                    &--there are voice packets waiting
                       BEGIN
                       IF (j <> super) AND (counter = 0) THEN
                       &--counter is 0 before coming back to supervisor
                          BEGIN
                          counter := CIV;
                          late := 1;
                          END;
                       SET(vtrans(j));
                       RESET(circ);
                       END;
                    j := j + 1 - INTREAL(j/N)*N;
                    &--move to next station
                    IF j = super THEN
                       BEGIN
                       IF late = 1 THEN
                       &--no data packets will be served.
                          BEGIN
                          priority := 1;
                          late := 0;
                          END
                       ELSE
                       &--start data round
                          priority := 2;
                       END;
                    TRANSIT(token);
                    END
                 ELSE
                 &--priority=2, token serves data packets
                    BEGIN
                    IF s(j).NB > 0 THEN
                    &--there are data packets waiting
```

```
                    BEGIN
                    SET(transmit(j));
                    RESET(circ);
                    END;
                 j := j + 1 - INTREAL(j/N)*N;
                 &--move to next station
                 IF j = super THEN
                 &--finish data round, start voice round
                    BEGIN
                    WAIT(circ);
                    counter := CIV;
                    priority := 1;
                    END;
                 TRANSIT (token);
                 END;
              END;


&------------------
& TMAX and printout
&------------------
/CONTROL/
     TMAX     =  150000;
     ACCURACY =  time, vpid, vs;
     ENTRY    =  BEGIN
                 PRINT(" ");
                 PRINT(" ");
                 PRINT("input parameters:");
                 PRINT(" number of stations", N);
                 PRINT(" voice call holding time", mholding);
                 PRINT(" voice station idle time", midle);
                 PRINT(" talkspurt last time", mlasting);
                 PRINT(" silent time", msilent);
                 PRINT(" voice packet generating time", vpgt);
                 PRINT(" data arrival rate(message/ms)", source(1).t);
                 PRINT(" message length", mleng);
                 PRINT(" packet length", pleng);
                 PRINT(" walk time between two stations", walk);
                 PRINT(" counter initial value", CIV);
                 PRINT(" ");
                 END;
     EXIT     =  BEGIN
                 PRINT("results");
                 PRINT(" mean data response for 1", MRESPONSE(time(1)));
                 PRINT(" mean data response for 4", MRESPONSE(time(4)));
                 PRINT(" mean data response for 7", MRESPONSE(time(7)));
                 PRINT(" mean data response for 10",MRESPONSE(time(10)));
                 PRINT(" mean voice response for 1", MRESPONSE(vs(1)));
                 PRINT(" mean voice response for 4", MRESPONSE(vs(4)));
                 PRINT(" mean voice response for 7", MRESPONSE(vs(7)));
                 PRINT(" mean voice response for 10",MRESPONSE(vs(10)));
                 PRINT("mean v.p. interdeparture for 1", MRESPONSE(vpid(1)));
                 PRINT("  variance of interdeparture", VRESPONSE(vpid(1)));
                 PRINT("mean v.p. interdeparture for 4", MRESPONSE(vpid(4)));
                 PRINT("  variance of interdeparture", VRESPONSE(vpid(4)));
     END;
```

```
&-----------------------------------------
& initialization, simulation, and calculation
&-----------------------------------------
$MACRO exec(arr, leng)
/EXEC/
     BEGIN
     FOR i := 1 STEP 1 UNTIL N DO
        BEGIN
        vs(i).id := i;
        server(i).id := i;
        vpkt(i).id := i;
        vcall(i).id := i;
        talkspur(i).id := i;
        vpid(i).id := i;
        s(i).id := i;
        pkt(i).id := i;
        source(i).id := i;
        source(i).t := arr;
        vcall(i).hook := 0;
        talkspur(i).silent := 1;
        END;
     super := 1;
     priority := 1;
     j := 1;
     mholding := 900.0;
     midle := 900.0;
     mlasting := 45.0;
     msilent := 36.0;
     vpgt := 3;
     mleng :=leng;
     pleng := 1;
     ptran := 0.2;
     walk := 0.001;
     counter := CIV;
     late := 0;
     FOR CIV := 5,8,10, 12, 15 DO
        BEGIN
        mrtv := 0;
        civ := 0;
        mrtd := 0;
        cid := 0;
        variance := 0;
        dvarian := 0;
        SIMUL;
        FOR i := 1 STEP 1 UNTIL N DO
           BEGIN
           mrtv := mrtv + (MRESPONSE(vs(i))+ptran)/N;
           civ := civ + CRESPONSE(vs(i))/N;
           mrtd := mrtd + MRESPONSE(time(i))/N;
           cid := cid + CRESPONSE(time(i))/N;
           variance := variance + VRESPONSE(vs(i))/N;
           dvarian := dvarian + VRESPONSE(time(i))/N;
           midt := midt + MRESPONSE(vpid(i))/N;
           vidt := vidt + VRESPONSE(vpid(i))/N;
           ciidt := ciidt + CRESPONSE(vpid(i))/N;
           END;
        PRINT(" mean response for voice", mrtv);
        PRINT(" variance of voice delay", variance);
```

```
        PRINT(" confidence interval for mrtv", 100*civ/mrtv, "%");
        PRINT(" mean response for data", mrtd);
        PRINT(" variance of data delay", dvarian);
        PRINT(" confidence interval for mrtd", 100*cid/mrtd, "%");
        PRINT(" mean interdeparture time for voice", midt);
        PRINT(" variance of interdeparture time for voice", vidt);
        PRINT(" confidence interval for midt", 100*ciidt/midt, "%");
        END;
    END;
$END


&------------------
& change parameters
&------------------
$exec(0.002, 10.0)
$exec(0.005, 10.0)
$exec(0.010, 10.0)
$exec(0.015, 10.0)
$exec(0.018, 10.0)
$exec(0.020, 10.0)
$exec(0.025, 10.0)


/END/
```

C. Simulation Code for the RCP-II

```
& This is the same as Appendix B, except token circulation is controlled
& differently. Compare QUEUE token.


/CONTROL/
   OPTION     =  NSOURCE;
/DECLARE/
   INTEGER
      N=10,          & number of stations on the network
      priority,      & the token is for voice (1) or data (2)
      super,         & the supervisor station
      R=1,           & counter for number of rotation rounds
      nbout,         & total number of voice packets generated in simulation
      bnbout,        & total number of data packets generated in simulation
      CIV,           & counter initial value (CIV)
      counter,       & counter value
      late = 0,      & the L bit in the token
      pnb(N),        & number of voice packets in the talkspurt
      dpnb,          & number of data packets in the message
      dataloss,      & nb of data packets lost when the network is saturated
      i, j;

   REAL
      mleng,         & mean data message length (number of packets)
      pleng,         & data packet length
      ptran,         & one packet transmission time (time unit)
      walk,          & walktime between two stations (time unit)
      mholding,      & mean voice call holding time (time unit)
      midle,         & mean voice call idle time (time unit)
      mlasting,      & mean talkspurt lasting time (time unit)
      msilent,       & mean silence period duration (time unit)
      vpgt,          & voice packet generation time (time unit)
      mrtv,          & mean voice packet delay (time unit)
      variance,      & variance of voice packets (square of time unit)
      civ,           & confidence interval of mrtv
      mrtd,          & mean data message delay (time unit)
      dvarian,       & variance of mrtd (square of time unit)
      cid,           & confidence interval of mrtd
      midt,          & mean voice packet interdeparture time (time unit)
      vidt,          & variance of midt (square of time unit)
      ciidt,         & confidence interval of midt
      holding(N),    & voice call holding time (time unit)
      last(N),       & talkspurt lasting time (time unit)
      tsilent(N),    & silence lasting time (time unit)
      x, y;

   QUEUE INTEGER
      id,            & index of station number
      hook,          & indicating status of on-hook(0)/off-hook (1)
      silent;        & indicating talkspurt(1)/silence(0)

   QUEUE REAL
     t;              & mean data message interarrival time
```

```
   QUEUE
      s(N),          & data stations
      pkt(N),        & packetization for data messages
      source(N),     & data messages arrival
      server(N),     & serving voice packets
      vs(N),         & voice stations
      vpkt(N),       & packetization for talkspurts
      vcall(N),      & generating voice calls
      talkspur(N),   & generating talkspurts
      vpid(N),       & voice packet interdeparture after transmission
      time(N),       & measuring data message response time
      token,         & token
      init;          & initial queue

   FLAG
      vtrans(N),     & control voice packet transmission
      transmit(N),   & control data packet transmission
      circ,          & control token circulation
      endcall(N),    & indicating voice call end
      interd(N);     & indicating voice packet departure

   CUSTOMER INTEGER
      SEQ;

   REF CUSTOMER
      C,
      cv;


/CONTROL/
   OPTION     =  NRESULT;
   MARGINAL   =  time, vpid, vs;

&-------------------------------------------------------
&  This is for generating voice call:
& voice call will last for sometime (mholding), and then
& this voice station will be idle for sometime (midle).
& Two states (holding and idle) will be alternative.
&-------------------------------------------------------
/STATION/
   NAME       =  vcall;
   TYPE       =  SOURCE;
   SERVICE    =  BEGIN
              IF vcall(id).hook=1 THEN
                 BEGIN
                 holding(id) := EXP(mholding);
                 CST(holding(id));
                 vcall(id).hook := 0;
                 END
              ELSE
                 BEGIN
                 RESET(endcall(id));
                 EXP(midle);
                 SET (endcall(id));
                 vcall(id).hook := 1;
                 END;
              TRANSIT(OUT);
```

```
            END;

&-------------------------------------------------------------
&  During each voice call holding time, there will be talkspurt
& and silent alternatively. Talkspurt will last for EXP(mlasting),
& then the following will be silent for EXP(msilent).
&-------------------------------------------------------------
/STATION/
   NAME       =  talkspur;
   TYPE       =  SOURCE;
   SERVICE    =  BEGIN
              IF vcall(id).hook = 1 THEN
                 BEGIN
                 IF talkspur(id).silent = 1 THEN
                    BEGIN
                    last(id) := EXP(mlasting);
                    IF last(id) < holding(id) THEN
                       holding(id) := holding(id)-last(id)
                    ELSE
                       last(id) := holding(id);
                    TRANSIT(NEW(CUSTOMER),vpkt(id));
                    CST(last(id));
                    talkspur(id).silent := 0;
                    END
                 ELSE
                    BEGIN
                    tsilent(id) := EXP(msilent);
                    IF tsilent(id) < holding(id) THEN
                       holding(id) := holding(id)-tsilent(id)
                    ELSE
                       tsilent(id) := holding(id);
                    CST(tsilent(id));
                    talkspur(id).silent := 1;
                    END;
                 END
              ELSE
                 BEGIN
                 talkspur(id).silent := 1;
                 WAIT(endcall(id));
                 END;
              TRANSIT(OUT);
              END;


&--------------------------
&   generating voice packets
&--------------------------
/STATION/
   NAME    =  vpkt;
   SERVICE    =  BEGIN
              y := last(id);
              pnb(id) := INTREAL(y/vpgt);
              IF pnb(id) > 0 THEN
                 FOR i := 1 STEP 1 UNTIL pnb(id) DO
                    BEGIN
                    CST(vpgt);
                    cv :=  NEW(CUSTOMER);
                    cv.SEQ :=0;
```

```
                TRANSIT(cv, vs(id));
                END;
            SEQ := 2;
            TRANSIT(vs(id));
            END;


&----------------
&  voice station
&----------------
/STATION/
   NAME    =  vs;
   SERVICE    =  BEGIN
            WAIT(vtrans(id));
            RESET(vtrans(id));
            TRANSIT(server(id));
            END;


&---------------------------
&  voice packet transmission
&---------------------------
/STATION/
   NAME    =  server;
   SERVICE    =  BEGIN
            CST(ptran);
            counter := counter - 1;
            SET(circ);
            SET(interd(id));
            RESET(interd(id));
            IF SEQ = 2 THEN
               BEGIN
               TRANSIT(OUT);
               END;
            TRANSIT(vpid(id));
            END;


&---------------------------------
&  voice packet interdeparture time
&---------------------------------
/STATION/
   NAME       =  vpid;
   SERVICE    =  BEGIN
                 WAIT(interd(id));
                 TRANSIT(OUT);
                 END;


&-----------------------------------------------
& It's for measuring the message response time
&  - from generation to finishing transmission
&  - by P/V procedure
&-----------------------------------------------
/STATION/
   NAME    =  time(1 STEP 1 UNTIL N);
   TYPE    =  RESOURCE, INFINITE;
```

```
&---------------------
&  generating messages
&---------------------
/STATION/
   NAME       =  source;
   TYPE       =  SOURCE;
   SERVICE    =  BEGIN
                 EXP(1./t);
                 P(time(id));
                 TRANSIT(pkt(id));
                 END;


&---------------
&  packetization
&---------------
/STATION/
   NAME    =  pkt;
   SERVICE    =  BEGIN
              x := EXP(mleng);
              dpnb := INTREAL(x / pleng);
              IF dpnb > 0 THEN
                 FOR i := 1 STEP 1 UNTIL dpnb DO
                    BEGIN
                    C := NEW(CUSTOMER);
                    C.SEQ := 0;
                    IF CUSTNB(s(id)) > 500 THEN
                       BEGIN
                       dataloss := dataloss + 1;
                       TRANSIT(C, OUT);
                       END
                    ELSE
                       TRANSIT (C, s(id));
                    END;
              SEQ := 1;
              TRANSIT(s(id));
              END;


&-------------
&  user station
&-------------
/STATION/
   NAME       =  s;
   SERVICE    =  BEGIN
              WAIT(transmit(id));
              RESET(transmit(id));
              CST(ptran);
              counter := counter - 1;
              IF (counter > 0) AND (s(id).NB > 1) THEN
                 BEGIN
                 SET(transmit(id));
                 END
              ELSE
                 BEGIN
                 IF counter = 0 THEN
                    BEGIN
```

```
                  R := 1;
                  counter := CIV;
                  priority := 1;
                  super := id + 1 - INTREAL(id/N)*N;
                  END;
               SET(circ);
               END;
            IF SEQ = 1 THEN
               V(time(id));
            TRANSIT(OUT);
            END;


&-------------------------
&  initial queue
&     - to set circ = true
&-------------------------
/STATION/
   NAME       =  init;
   INIT       =  1;
   SERVICE    =  SET (circ);
   TRANSIT    =  OUT;


&-----------------------------------------------------------------------
&  token circulating in ring
&     This is based on RCP-II MAC protocol. See Chapter 3 for details
&-----------------------------------------------------------------------
/STATION/
   NAME       =  token;
   INIT       =  1;
   SERVICE    =
      BEGIN
      WAIT(circ);
      CST(walk);
      IF priority = 1 THEN
      &--token serves voice packets
          BEGIN
          IF vs(j).NB >0 THEN
          &--there are voice packets waiting
             BEGIN
             IF (j <> super) AND (counter = 0) THEN
             &--counter is 0 before coming back to supervisor
                BEGIN
                counter := CIV;
                late := 1;
                END;
             SET(vtrans(j));
             RESET(circ);
             END;
          j := j + 1 - INTREAL(j/N)*N;
          IF j = super THEN
             BEGIN
             WAIT(circ);
             IF (counter>0) AND (late=0) THEN
             &--start data round
                BEGIN
                priority := 2;
```

```
            END
         ELSE
         &--no time left for data round
            BEGIN
            late := 0;
            IF counter=0 THEN
               BEGIN
               counter := CIV;
               END;
            END;
         END;
      TRANSIT(token);
      END
   ELSE
   &--token serves data packets
      BEGIN
      IF s(j).NB > 0 THEN
      &--there are data packets waiting
         BEGIN
         SET(transmit(j));
         RESET(circ);
         END;
      j := j + 1 - INTREAL(j/N)*N;
      IF j = super THEN
         BEGIN
         WAIT(circ);
         &--if token walking time is longer than packet transmission
         &--time, the counter should be decremented.
         counter := counter - INTREAL((walk*N*(R+1))/ptran);
         R := R + 1 - INTREAL((walk*N*(R+1))/ptran)*(ptran/(walk*N));
         IF counter=0 THEN  &--end data round and start voice round
            BEGIN
            counter := CIV;
            priority := 1;
            END;
         END;
         &--else, priority=2, token will serve data packets again
      TRANSIT (token);
      END;
   END;


&-----------------
& TMAX and printout
&-----------------
/CONTROL/
   TMAX       =  150000;
   ACCURACY   =  time, vpid, vs;
   ENTRY      =  BEGIN
                 PRINT(" ");
                 PRINT(" ");
                 PRINT("input parameters:");
                 PRINT(" number of stations", N);
                 PRINT(" voice call holding time", mholding);
                 PRINT(" voice station idle time", midle);
                 PRINT(" talkspurt last time", mlasting);
                 PRINT(" silent time", msilent);
                 PRINT(" voice packet generating time", vpgt);
```

```
                PRINT(" data arrival rate(message/ms)", source(1).t);
                PRINT(" message length", mleng);
                PRINT(" packet length", pleng);
                PRINT(" walk time between two stations", walk);
                PRINT(" counter initial value", CIV);
                PRINT(" ");
                END;
   EXIT      =  BEGIN
                PRINT("results");
                PRINT(" mean data response for 1",MRESPONSE(time(1)));
                PRINT(" mean data response for 4",MRESPONSE(time(4)));
                PRINT(" mean voice response for 1",MRESPONSE(vs(1)));
                PRINT(" mean voice response for 4",MRESPONSE(vs(4)));
                PRINT("mean v.p. interdeparture for 1", MRESPONSE(vpid(1)));
                PRINT("  variance of interdeparture", VRESPONSE(vpid(1)));
                PRINT("mean v.p. interdeparture for 4", MRESPONSE(vpid(4)));
                PRINT("  variance of interdeparture", VRESPONSE(vpid(4)));
                END;


&-----------------------------------------
& initialization, simulation, and calculation
&-----------------------------------------
$MACRO exec(arr, leng)
/EXEC/
   BEGIN
   FOR i := 1 STEP 1 UNTIL N DO
      BEGIN
      vpid(i).id := i;
      server(i).id := i;
      vs(i).id := i;
      vpkt(i).id := i;
      vcall(i).id := i;
      talkspur(i).id := i;
      s(i).id := i;
      pkt(i).id := i;
      source(i).id := i;
      source(i).t := arr;
      vcall(i).hook := 0;
      talkspur(i).silent := 1;
      END;
   super := 1;
   priority := 1;
   j := 1;
   mholding := 900.0;
   midle := 900.0;
   mlasting := 45.0;
   msilent := 36.0;
   ptran := 0.2;
   mleng :=leng;
   pleng := 1;
   walk := 0.001;
   counter := CIV;
   FOR CIV := 15  DO
      BEGIN
      vpgt := CIV*ptran;
      mrtv := 0;
      civ := 0;
```

```
      mrtd := 0;
      cid := 0;
      variance := 0;
      dvarian := 0;
      midt := 0;
      vidt := 0;
      ciidt := 0;
      nbout := 0;
      bnbout := 0;
      dataloss := 0;
      SIMUL;
      FOR i := 1 STEP 1 UNTIL N DO
           BEGIN
           mrtv := mrtv + (MRESPONSE(vs(i))+ptran)/N;
           civ := civ + CRESPONSE(vs(i))/N;
           mrtd := mrtd + MRESPONSE(time(i))/N;
           cid := cid + CRESPONSE(time(i))/N;
           variance := variance + VRESPONSE(vs(i))/N;
           nbout := nbout + vs(i).NBOUT;
           dvarian := dvarian + VRESPONSE(time(i))/N;
           midt := midt + MRESPONSE(vpid(i))/N;
           vidt := vidt + VRESPONSE(vpid(i))/N;
           ciidt := ciidt + CRESPONSE(vpid(i))/N;
           bnbout := bnbout + s(i).NBOUT;
           END;
      PRINT(" mean response for voice", mrtv);
      PRINT(" variance of voice delay", variance);
      PRINT(" confidence interval for mrtv", 100*civ/mrtv, "%");
      PRINT("  mean response for data", mrtd);
      PRINT("  variance of data delay", dvarian);
      PRINT("  confidence interval for mrtd", 100*cid/mrtd, "%");
      PRINT(" mean interdeparture time for voice", midt);
      PRINT(" variance of interdeparture time for voice", vidt);
      PRINT(" confidence interval for midt", 100*ciidt/midt, "%");
      PRINT(" nb of data loss", dataloss);
      PRINT(" total nb of voice packets", nbout);
      PRINT(" total nb of data packets", bnbout+dataloss);
      END;
   END;
$END


&--------------------
&  change parameters
&--------------------
$exec(0.002, 10)
$exec(0.010, 10)
$exec(0.020, 10)
$exec(0.030, 10)
$exec(0.033, 10)
$exec(0.035, 10)
$exec(0.045, 10)


/END/
```

D. Simulation Code for the finite buffering scheme

```
/CONTROL/
    OPTION    =  NSOURCE;

/DECLARE/
    INTEGER
       N=10,            & station number
       M=4,             & voice arrival nb per station
       priority,        & the token is for voice (1) or data (2)
       super,           & supervisor station
       R=0,             & rotation round counter
       nbout,           & total number of voice packet served
       CIV=40,          & counter initial value(CIV)
       counter,         & counter value
       late = 0,        & the L bit in the token
       buffer(N),       & number of voice packets in buffer
       pnb(N, M),       & number of voice packets
       dpnb,            & number of data packets
       BufferSize,      & buffer size limit
       i, i1, j, k, m, m1, m2;

    REAL
       mleng,           & data message length
       pleng,           & data packet length
       ptran,           & packet transmission time
       walk,            & walk time between two stations
       mlasting,        & mean talkspurt lasting time
       msilent,         & mean silent time
       vpgt,            & voice packet generating time
       mrtv,            & mean response time for voice
       mrtd,            & mean response time for data
       civ,             & confidence interval for mrtv
       cid,             & confidence interval for mrtd
       variance,        & variance of mrtv
       dvarian,         & variance for mrtd
       midt,            & mean voice packet interdeparture time
       vidt,            & variance of midt
       ciidt,           & confidence interval of midt
       x, y(N, M);

    QUEUE INTEGER
       id,              & index of station number
       mult,            & index of voice calls per station
       silent;          & indicating talkspurt(1)/silent(0)

    QUEUE REAL
       t;               & mean data message interarrival time

    QUEUE
       s(N),            & data stations
       pkt(N),          & packetization for data messages
       source(N),       & data messages arrival
```

```
        server(N),      & serving voice packets
        loss,           & collect all lost packets
        vs(N, M),       & voice stations
        vpkt(N, M),     & packetization for talkspurts
        talkspur(N, M),& generating talkspurts
        vpid(N, M),     & voice packet interdeparture after transmission
        time(N),        & measuring data message response time
        token,          & token
        init;           & initial queue

    FLAG
        vtrans(N, M),   & control voice packet transmission
        transmit(N),    & control data packet transmission
        circ,           & control token circulation
        interd(N, M);   & indicating voice packet departure

    CUSTOMER INTEGER
        SEQ;

    REF CUSTOMER
        C,
        cv;

/CONTROL/
    OPTION    =  NRESULT;
    MARGINAL =  time, vpid, vs;


&-----------------------------------------------------------------
& During each voice call holding time, there will be talkspurt
& and silent alternatively. Talkspurt will last for EXP(mlasting),
& then the following will be silent for EXP(msilent).
&-----------------------------------------------------------------
/STATION/
    NAME      =  talkspur;
    TYPE      =  SOURCE;
    SERVICE   =
        BEGIN
        IF talkspur(id, mult).silent = 1 THEN
            BEGIN
            TRANSIT(NEW(CUSTOMER),vpkt(id, mult));
            y(id, mult) := EXP(mlasting);
            CST(y(id, mult));
            talkspur(id, mult).silent := 0;
            END
        ELSE
            BEGIN
            EXP(msilent);
            talkspur(id, mult).silent := 1;
            END;
        TRANSIT(OUT);
     END;

&-------------------------
& generating voice packets
&-------------------------
/STATION/
    NAME      =  vpkt;
```

```
    SERVICE  =
       BEGIN
       UNIFORM (0.0, vpgt);
       pnb(id, mult) := INTREAL(y(id, mult)/vpgt);
       IF pnb(id, mult) > 0 THEN
          FOR i1 := 1 STEP 1 UNTIL pnb(id, mult) DO
             BEGIN
             CST(vpgt);
             cv :=  NEW(CUSTOMER);
             cv.SEQ :=0;
             TRANSIT(cv, vs(id, mult));
             END;
       SEQ := 2;
       TRANSIT(vs(id, mult));
       END;


&-------------------------------------------------------------------------
&  voice station
&    this is an example of finite buffer. The buffer is shared by all voice
&    sources on station i. If total waiting voice packets exceeds BufferSize,
&    the coming voice packet will be lost(go to QUEUE loss)
&-------------------------------------------------------------------------
/STATION/
   NAME       =  vs;
   SERVICE    =
      BEGIN
      FOR m := 1 STEP 1 UNTIL M DO
           BEGIN
           buffer(id) := CUSTNB(vs(id, m))+ buffer(id);
           END;
      IF buffer(id) > BufferSize THEN
           BEGIN
           buffer(id) := 0;
           TRANSIT(loss);
           END;
      buffer(id) := 0;
      WAIT(vtrans(id, mult));
      RESET(vtrans(id, mult));
      TRANSIT(server(id, mult));
      END;


&--------------------------
&  voice packet transmission
&--------------------------
/STATION/
   NAME    =  server;
   SERVICE    =
        BEGIN
        CST(ptran);
        counter := counter - 1;
        SET(circ);
        SET(interd(id, mult));
        RESET(interd(id, mult));
        IF SEQ = 2 THEN
           BEGIN
           TRANSIT(OUT);
```

```
         END;
      TRANSIT(vpid(id, mult));
      END;


&---------------------------------
& voice packet interdeparture time
&---------------------------------
/STATION/
    NAME     =  vpid;
    SERVICE  =
       BEGIN
       WAIT(interd(id, mult));
       TRANSIT(OUT);
       END;


&----------------
& voice call lost
&----------------
/STATION/
    NAME     =  loss;
    SERVICE  =  TRANSIT(OUT);


&-------------------------------------------
& It's for measuring the message response time
&    - from generation to finishing transmission
&    - by P/V procedure
&-------------------------------------------
/STATION/
    NAME     =  time(1 STEP 1 UNTIL N);
    TYPE     =  RESOURCE, INFINITE;


&--------------------
& generating message
&--------------------
/STATION/
    NAME     =  source;
    TYPE     =  SOURCE;
    SERVICE  =
       BEGIN
       EXP(1./t);
       P(time(id));
       TRANSIT(pkt(id));
       END;

&--------------
& packetization
&--------------
/STATION/
    NAME     =  pkt;
    SERVICE  =
       BEGIN
       x := EXP(mleng);
       dpnb := INTREAL(x / pleng);
       IF dpnb > 0 THEN
```

```
            FOR i1 := 1 STEP 1 UNTIL dpnb DO
               BEGIN
               C := NEW(CUSTOMER);
               C.SEQ := 0;
               TRANSIT (C, s(id));
               END;
         SEQ := 1;
         TRANSIT(s(id));
         END;


&-------------
& user station
&-------------
/STATION/
     NAME     =  s;
     SERVICE  =
        BEGIN
        WAIT(transmit(id));
        RESET(transmit(id));
        CST(ptran);
        counter := counter - 1;
        IF (counter > 0) AND (s(id).NB > 1) THEN
           BEGIN
           SET(transmit(id));
           END
        ELSE
           BEGIN
           IF counter = 0 THEN
              BEGIN
              R := 1;
              counter := CIV;
              priority := 1;
              super := id + 1 - INTREAL(id/N)*N;
              END;
           SET(circ);
           END;
        IF SEQ = 1 THEN
           V(time(id));
        TRANSIT(OUT);
        END;



&-----------------------------------
& initial queue
&    - to set circ = true
&-----------------------------------
/STATION/
     NAME     =  init;
     INIT     =  1;
     SERVICE  =  SET (circ);
     TRANSIT  =  OUT;

&-------------------------------------------------------------------
& token circulating in ring
&    -- it is based on RCP-II MAC protocol. See Chapter 3 for details
&-------------------------------------------------------------------
```

```
/STATION/
    NAME     =  token;
    INIT     =  1;
    SERVICE  =
       BEGIN
       WAIT(circ);
       CST(walk);
       IF priority = 1 THEN
       &--serve voice packets
          BEGIN
          FOR m1 := 1 STEP 1 UNTIL M DO
          &--transmit voice packets from each voice source
             BEGIN
             WAIT(circ);
             IF vs(j, m1).NB >0 THEN
                BEGIN
                IF (j <> super) AND (counter = 0) THEN
                   BEGIN
                   counter := CIV;
                   late := 1;
                   END;
                SET(vtrans(j, m1));
                RESET(circ);
                END;
             END;
          j := j + 1 - INTREAL(j/N)*N;
          IF j = super THEN
             BEGIN
             WAIT(circ);
             R := R+1;
             IF (counter>0) AND (late=0) THEN
             &--start data round
                BEGIN
                priority := 2;
                END
             ELSE &--counter=0 or late=1
             &--no time left for data round
                BEGIN
                late := 0;
                IF counter=0 THEN
                   BEGIN
                   counter := CIV;
                   END;
                END;
             END;
          TRANSIT(token);
          END
       ELSE
       &--priority=2, serve data packets
          BEGIN
          IF s(j).NB > 0 THEN
          &--there are data packets waiting
             BEGIN
             SET(transmit(j));
             RESET(circ);
             END;
          j := j + 1 - INTREAL(j/N)*N;
          IF j = super THEN
```

```
            BEGIN
            WAIT(circ);
            &--if token walking time is longer than packet transmission
            &--time, the counter should be decremented.
            counter := counter-INTREAL((walk*N*R)/ptran);
            R := R + 1 - (ptran/(walk*N))*INTREAL((walk*N*R)/ptran);
            IF counter<=0 THEN
               BEGIN
               counter := CIV+counter;
               priority := 1;
               END;
            END;
         TRANSIT (token);
         END;
      END;


&--------------------
& TMAX and printout
&--------------------
/CONTROL/
     TMAX     =  150000;
     ACCURACY =  time, vs, vpid;
     ENTRY    =  BEGIN
        PRINT(" ");
        PRINT(" ");
        PRINT("input parameters:");
        PRINT(" number of stations", N);
        PRINT(" number of arrivals per station", M);
        PRINT(" buffer size", BufferSize);
        PRINT(" talkspurt last time", mlasting);
        PRINT(" silent time", msilent);
        PRINT(" voice packet generating time", vpgt);
        PRINT(" data arrival rate(message/ms)", source(1).t);
        PRINT(" message length", mleng);
        PRINT(" packet length", pleng);
        PRINT(" walk time between two stations", walk);
        PRINT(" counter initial value", CIV);
        PRINT(" ");
        END;
     EXIT     =  BEGIN
        PRINT("results");
        PRINT(" mean data response for 1", MRESPONSE(time(1)));
        PRINT(" mean data response for 4", MRESPONSE(time(4)));
        PRINT(" mean data response for 7", MRESPONSE(time(7)));
        PRINT(" mean data response for 10",MRESPONSE(time(10)));
        PRINT(" mean voice response for 1", MRESPONSE(vs(1,1)));
        PRINT(" mean voice response for 4", MRESPONSE(vs(4,1)));
        PRINT(" mean voice response for 7", MRESPONSE(vs(7,1)));
        PRINT(" mean voice response for 10",MRESPONSE(vs(10,1)));
        PRINT("mean v.p. interdeparture for 1,1", MRESPONSE(vpid(1, 1)));
        PRINT("mean v.p. interdeparture for 1,2", MRESPONSE(vpid(1, 2)));
        PRINT("mean v.p. interdeparture for 1,3", MRESPONSE(vpid(1, 3)));
        PRINT("  variance of interdeparture", VRESPONSE(vpid(1, 1)));
        PRINT("mean v.p. interdeparture for 4", MRESPONSE(vpid(4, 1)));
        PRINT("  variance of interdeparture", VRESPONSE(vpid(4, 1)));
        PRINT("mean v.p. interdeparture for 7", MRESPONSE(vpid(7, 1)));
        PRINT("  variance of interdeparture", VRESPONSE(vpid(7, 1)));
```

```
        PRINT("mean v.p. interdeparture for 10", MRESPONSE(vpid(10, 1)));
        PRINT("  variance of interdeparture", VRESPONSE(vpid(10, 1)));
        END;


&-----------------------------------------
& initialization, simulation, and calculation
&-----------------------------------------
$MACRO exec(arr, leng)
/EXEC/
     BEGIN
     FOR i := 1 STEP 1 UNTIL N DO
        BEGIN
        FOR m := 1 STEP 1 UNTIL M DO
           BEGIN
           vpid(i,m).id := i;
           vs(i, m).id := i;
           server(i, m).id := i;
           vpkt(i, m).id := i;
           talkspur(i, m).id := i;
           vpid(i,m).mult := m;
           vs(i, m).mult := m;
           vpkt(i, m).mult := m;
           talkspur(i, m).mult := m;
           talkspur(i, m).silent := 1;
           END;
        s(i).id := i;
        pkt(i).id := i;
        source(i).id := i;
        source(i).t := arr;
        END;
     FOR BufferSize := 5, 8, 12 DO
        BEGIN
        super := 1;
        priority := 1;
        j := 1;
        m := 1;
        mlasting := 45.0;
        msilent := 36.0;
        ptran := 0.2;
        mleng :=leng;
        pleng := 1;
        walk := 0.001;
        counter := CIV;
        nbout := 0;
        vpgt := CIV*ptran;
        mrtv := 0;
        civ := 0;
        mrtd := 0;
        cid := 0;
        variance := 0;
        dvarian := 0;
        midt := 0;
        vidt := 0;
        ciidt := 0;
        SIMUL;
        FOR i1 := 1 STEP 1 UNTIL N DO
           BEGIN
```

```
            FOR m2:= 1 STEP 1 UNTIL M DO
               BEGIN
               mrtv := mrtv + (MRESPONSE(vs(i, m))+ptran)/(N*M);
               variance := variance + VRESPONSE(vs(i, m))/(N*M);
               civ := civ + CRESPONSE(vs(i, m))/(N*M);
               midt := midt + MRESPONSE(vpid(i, m))/(N*M);
               vidt := vidt + VRESPONSE(vpid(i, m))/(N*M);
               ciidt := ciidt + CRESPONSE(vpid(i, m))/(N*M);
               nbout := nbout + vs(i, m).NBIN;
               END;
            mrtd := mrtd + MRESPONSE(time(i))/N;
            cid := cid + CRESPONSE(time(i))/N;
            dvarian := dvarian + VRESPONSE(time(i))/N;
            END;
         PRINT(" mean response for voice", mrtv);
         PRINT(" variance of voice delay", variance);
         PRINT(" confidence interval for mrtv", 100*civ/mrtv, "%");
         PRINT("  mean response for data", mrtd);
         PRINT("  variance of data delay", dvarian);
         PRINT("  confidence interval for mrtd", 100*cid/mrtd, "%");
         PRINT(" mean interdeparture time for voice", midt);
         PRINT(" variance of interdeparture time for voice", vidt);
         PRINT(" confidence interval for midt", 100*ciidt/midt, "%");
         PRINT(" nb of loss", loss.NBOUT);
         PRINT(" nb of voice packets", nbout);
         PRINT(" loss probability", 100*loss.NBOUT/nbout,"%");
         END;
      END;
$END

&--------------------
& change parameters
&--------------------
$exec(0.002, 10)
$exec(0.005, 10)
$exec(0.010, 10)
$exec(0.012, 10)
$exec(0.015, 10)
$exec(0.017, 10)

/END/
```