New Communication Network Protocol for a Data Acquisition System

Tomohisa Uchida

DOCTOR OF PHILOSOPHY

Department of Particle and Nuclear Physics,
School of High Energy Accelerator Science,
The Graduate University for Advanced Studies

2004

New Communication Network Protocol for a Data Acquisition System

By

Tomohisa Uchida

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in the

The Graduate University for Advanced Studies

Committee in charge:

Prof. Junsei Chiba, Chair

Prof. Hirofumi Fujii

Prof. Hirokazu Ikeda

Prof. Nobuhiko Katayama

Prof. Yasushi Nagasaka

Prof. Manobu Tanaka

2004

New Communication Network Protocol for a Data Acquisition System

ABSTRACT

New Communication Network Protocol for a Data Acquisition System

By

Tomohisa Uchida
Doctor of Philosophy

The graduate university for advanced studies

Professor Junsei Chiba, Chair

Many high energy physics experiments have been searching a new physics beyond the standard model and it requires more precise detection and analyzing system than the current system. It requires many channels of the detectors, a large data size of an event, and a high rate trigger for a detector system. Since the amount of data handled in the system has increased, the required throughput for a data acquisition (DAQ) system is increased as well.

In order to satisfy the requirements, many DAQ systems adopt a distributed computer system and processes two or more events at the same time. A typical system consists of detector sub-systems, an event builder, event processors, and mass storage devices. The detector subsystem processes signals from detectors and sends the event fragment data to the event builder. The event builder collects these event fragment data from many detector sub-systems and builds completion events from it. The event data are sent to event processor. Finally the data are recorded on mass storage devices. Since an event builder communicates between many distributed detector subsystems, its communication technology is essential. Many recent DAQ systems have adopted network technologies for it, which are called network-based DAQ system. Various networks have been adopted, for example, IEEE 802.3 (Ethernet), asynchronous transfer mode (ATM), and so on. In particular, Ethernet is widely used because it is very cost effective infrastructure. A network such as Ethernet is used with network protocols because a network does not have a mechanism for reliable data delivery.

Since the TCP/IP protocol suite is a standard protocol suite of standard operating systems, for example, Linux, UNIX, and so on, many systems adopt it for reliable data delivery. When we employ the standard reliable protocols such as the TCP/IP protocol suite for an event builder, we encounter a serious problem: event fragments are heading for the same destination at the same time so that data flows are congested and packets are lost in the network. The packet losses decrease transfer efficiency and induce re-transmissions for reliable data delivery. Generally the mechanics of packet losses by congestion is too complex to predict its behavior and performance. Moreover, the re-transmission mechanism makes more complex situations. In order to solve the problem, there is a method in general networks. That is called a Quality of Service (QoS). There are various methods but the main idea is to assign bandwidths of connections and special

network devices are used for assigning bandwidths. There are two assignment methods: one assigns a fix bandwidth known as a constant-bit-rate (CBR) method, the other one dynamically assigns bandwidths known as a variable-bit-rate (VBR) method. If transfer rates of DAQ systems are constant, the CBR method solves the problem. Therefore, it is difficult to achieve a high efficiency with the CBR method. On the other hand, the VBR method requires additional protocols to control network devices for bandwidth assignments. Since the protocol assigns a bandwidth, it is also difficult to achieve a high efficiency because the protocol is not able to quickly control the devices.

When we design a highly efficient network, there is another problem which is packet losses by congestion. The packet losses make difficulties with prediction of transfer performance, and, then, a quantitatively network design. Since an even builder is designed for satisfying requirements of an experiment, this is a serious problem. It is rare case in general networks that many senders transmit to the same receiver at same time. Therefore, general reliable protocols are not designed for this case and we can not find a suitable protocol among standard ones. The other problem of using standard protocols is a heavy workload. If we need a high performance data transfer, we should adopt high performance hardware, for example, a high speed CPU, and so on.

In order to solve these problems, we have developed a new communication network protocol. The main idea of the new protocol is an avoiding congestion with a token passing mechanism. Since senders are controlled by the mechanism and only one sender that has a token is permitted to transmit data to a receiver, packet losses are avoided. By this mechanism, we can quantitatively design the system because its difficulty comes from a complicated mechanism of packet losses by congestion.

We introduced a sliding window mechanism to guarantee for reliable data delivery. The mechanism has a reliable data delivery as well as a data flow control between a sender and a receiver. In the mechanism, data transfer is controlled by an acknowledgement that is used for confirming of data transferred, and a sender transmits multiple packets before waiting for an acknowledgement, so the data are transferred in high efficiency.

We implemented the protocol and constructed systems to measure its performance. Since the protocol has a light workload, we implemented the protocol on a small hardware device to demonstrate that. We constructed various systems with the implementation on Ethernet. Since the protocol requires only packet switching function, we were able to use Ethernet hub and facilely constructed those system.

We employ a polling model in queuing theory for a mathematical model of the protocol. But since the model is mathematically simplified, we modified the equations of the model for our systems.

We measured the systems and analyzed these results with the mathematical model. We found that senders fairly transfer data among senders and the total bandwidth is above 90% from the measured results of transfer data rate variations. Since the calculated average message lengths were good agreement with measured results, we can predict it with given an average transfer data rate of senders and an average message waiting-times can be calculated.

We also measured and analyzed various systems which have different number of senders and network topologies. From the results, we found that these system performances can be calculated and we can quantitatively design a large scale network system. And we found that the protocol was more suitable for DAQ systems than TCP with comparison them.

From these results, the protocol is suitable for the DAQ systems and we can conclude that we can quantitatively design and construct a high performance DAQ system with the new protocol.

Table of Contents

Acknowledgments

This work has its roots in the teaching, help, patience, and inspiration of a great number of people, to whom I wish to express my heartfelt gratitude.

I wish to thank Prof. Hirofumi Fujii for supervising me and giving me a lot of advices on this research to develop a communication network protocol for a data acquisition system.

I would like to express my special gratitude to Prof. Manobu Tanaka. He gave me a wonderful opportunity to study this thesis in the Graduate University for Advanced studies. He patiently listened to my half-baked ideas of how to do and gave me a lot of advices.

I also wish to thank Prof. Yasushi Nagasaka who is with Hiroshima Institute of Technology. He gave me a lot of advices which greatly helps me.

I would like to also appreciate the on-line and electronics groups at High Energy Accelerator Research Organization (KEK) for their much sincerely support.

Finally, I would like to mention that this work would never have been realized without the continuous support provided by the High Energy Accelerator Research Organization during I am with the Graduate University for Advanced Studies.

正　誤　表

i ページ

誤：3.1.1 Entity.....20
　　3.1.2 Network.....21
　　3.1.3 Switch.....22

正：3.2.1 Entity.....20
　　3.2.2 Network.....21
　　3.2.3 Switch.....22

20 ページ

誤：3.1 Assumed configuration

正：3.2 Assumed configuration

誤：3.1.1 Entity

正：3.2.1 Entity

21 ページ

誤：3.1.2 Netwok

正：3.2.2 Network

22 ページ

誤：3.1.3 Switch

正：3.2.3 Switch

26 ページ

誤：3.3.1 Data transfer

正：3.3.2 Data transfer

Chapter 1

Introduction

High energy physics (HEP) is a science and studying the fundamental structure of matter and the ultimate constituents of matter as well as the nature of the interactions between them. Its ultimate aim is to find a complete description of the elementary constituents of matter and of the forces acting between them, and, then, the description should be as simple as possible.

The standard model (SM) appeared over twenty years ago and has well explained experimental results but the theory contains many free parameters and has some problems. Therefore, the theory is not considered for the ultimate theory and many experiments have been searching a new physics beyond the SM. For example, the precise tests of the SM are on going still at present. The BELLE [1] and BaBar [2] experiments are searching the CP-violation at High Energy Accelerator Research Organization (KEK) in Japan and Stanford Linear Accelerator Center (SLAC) in USA, respectively. The K2K [3] experiment is searching neutrino oscillations and a long baseline neutrino oscillation experiment at KEK and Institute for Cosmic Ray Research, University of Tokyo in Japan. Recently these experiments report that they find the signals of beyond the SM. In order to confirm evidences, next generation experiments are planned to construct a very large detector. At the European Organization for Nuclear Research (CERN) in Switzerland, the Large Hadron Collider (LHC) program [4] is in progress, where four detectors are been constructed; the ALICE [5], ATLAS [6], CMS [7], and LHCb [8] experiments. These experiments aim for searching and studying of quark-gluon-plasma, Higgs bosons, supersymmetry, CP-violation. The SupterB project [9] at KEK is planed to test SM more precisely.

In HEP experiment, to observe phenomena at sub-atomic and sub-nuclear levels, high energy particles are collided with a particle accelerator and the phenomena

are observed with a special detector system. The detector system consists of a lot of various type detectors, a trigger system, a data acquisition (DAQ) system, and a computing system. The detectors (for example, silicon vertex detectors, particle tracking detectors, particle identifiers) generate electronic signals by particles. The signals are sent to the trigger system that decides if the event is interesting. If the event is interesting then the detector signals are sent to the data acquisition system and a complete event data is built from these signals. The event data is sent to the computing system that eliminates those events with physics events, reconstructs results from data, and records results into storage devices. The BELLE detector system [10], for example, consists of eight detector types, from inside to out side, the silicon vertex detector, the central drift chamber, the aero-gel Cerenkov counter, the time-of-flight counter, the electromagnetic calorimetry, the K_L and muon detection system, and the extreme forward calorimeter. The total number of channels of the BELLE detector system is 150,000 and the required throughput for the DAQ system is 160 Mbit/sec.

Since a HEP experiment becomes more and more precise, it requires many channels of the detectors, a large data size of an event, and a high rate trigger for a detector system [5-9]. Therefore the total amount of data handled in the system is growing as well as the data size for a single event. In other words, the experiment requires a high throughput for a DAQ system because the main task of the system is to build an event data from a lot of event fragment data which are produced by many detectors and the system sends these data to a computing system. In the next experiments, very high data rates expected and the aggregated bandwidths required for event building are expected to be of the order 10-100 Gbit/sec [5-9], for example, over 16 Gbit/sec is required for the system at the SuperB experiment [9], over 500 Gbit/sec is required at the experiments of the LHC program [5-8].

Required throughputs for the DAQ systems have been increasing. It was few hundreds Kbit/sec in the 1970s, at the present time, it is few hundreds Mbit/sec and the next experiments are require a few hundreds Gbit/sec. In the 1970s, the DAQ systems gathered associated a single event to be processed in an online/offline fashion. In the 1980s and after, since the required throughput was increased, many systems adopted parallel processing for event buildings. In the parallel processing system, two or more events were built at the same time, and, then, the throughput could be increased. In

order to deal with the parallel processing system, the DAQ system has been organized under a subsystem of an event builder. The subsystem is a distributed computer system to acquire/eliminate events. Early subsystems adopted bus-based technologies for communicating between computers. At present time, in order to deal with increased data, many systems adopt communication network technologies, which are called network-based DAQ systems and various networks are adopted, for example, Asynchronous Transfer Mode (ATM) [11], IEEE802.3 (Ethernet) [12], and so on. The main reasons of adopting network are a high performance and a high flexibility. In particular, the Ethernet network is widely adopted [13] [14] [15] because its infrastructure is very cost effective [14].

A network requires a network protocol for data transfer. There are various protocols, which have been used. In particular, the TCP/IP protocol suite [16] is adopted by many systems [13] [15] [17] because it is a de-facto standard of standard OS's and has widely used. When we employ the standard reliable protocols such as the TCP/IP protocol suite for a DAQ system, we encounter a serious problem: event fragments are heading for same destination at the same time so that data flows are congested and a packet is lost in the network. The packet losses decrease transfer efficiency and induce re-transmissions for reliable data delivery. Since the re-transmission behavior of standard reliable protocols is complex, a transfer performance prediction is to be difficult. Therefore, we can not quantitatively design a DAQ system. It is the other problem of TCP that the protocol requires a heavy load for a CPU [17]. Therefore if we need to transfer in a high rate, we should use a high speed CPU.

In order to solve these problems, we developed a new communication network protocol. The new protocol is designed to keep the sequence of senders with a simple mechanism so that a packet loss is avoided and it requires a light load for a processor.

In this thesis, the new communication network protocol for a data acquisition system is described in detail. The outline of this thesis is as follows. In the chapter 2, we begin with discussion a network based DAQ system and point out a need of a new protocol development. In the chapter 3, a concept and mechanisms of the new protocol are described. In order to evaluate performance of the protocol, we implement the protocol and construct a test-bed system. In chapter 4, the implementation is described.

In chapter 5, the performance analysis is described and we will evaluate it. In chapter 6, extensions of the protocol are described. Finally, the summary is given in chapter 7.

References

[1] The BELLE Collaboration, "Letter of intent for a Study of CP Violation in B Meson Decays", KEK report 94-2, 1994.

[2] BaBar collaboration, "BaBar technical design report", SLAC-R-95-457, 1995.

[3] K. Nishikawa *et al.*, "Present and future neutrino oscillation experiments in Japan", Nucl.Phy.B (Proc. Suppl.), 59, 289, 1997.

[4] The LHC project, "The Large Hadron Collider", CERN/AC/95-05, 1995.

[5] The ALICE collaboration, "ALICE Technical proposal", CERN/LHCC 95-71, 1997.

[6] The ATRAS collaboration, "ATLAS Technical proposal", CERN/LHCC 94-43, 1994.

[7] The CMS collaboration, "The Compact Muon Solenoid, Technical proposal", CERN/LHCC 94-38, 1994.

[8] The LHCb collaboration, "LHCb Technical proposal", CERN/LHCC 98-004, 1998.

[9] K. Abe *et al.*, "Letter of Intent for KEK Super B Factory", KEK report 2004-04, 2004.

[10] The BELLE Collaboration, "The BELLE Detector", KEK progress report 2000-4, 2000.

[11] Oliver. Ibe, Essentials of ATM networks and Services, Addison Weslay, 1997.

[12] Institute of Electrical and Electronics Engineers (IEEE), "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE standard 802.3, 2003.

[13] T.J Pavel *et al.*, "Network Performance Testing for the BaBar Event Builder", Proceedings of the CHEP'98 Conference, 1998.

[14] S. Stancu *et al.*, "The use of Ethernet in the Data Flow of the ATLAS Trigger & DAQ", Proceedings of the CHEP'03, 2003.

[15] M. Nakao *et al.*, "Switchless Event Building Farm for the BELLE Data Acquisition System", IEEE Trans. on Nuclear Science, 48, 2385-2390, 2001.

[16] W. R. Stevens, "TCP/IP Illustrated, Volume 1: The protocols", Addison Weslay, 1994.

[17] R.W. Dobinson *et al.*, "IEEE 802.3 Ethernet, Current Status and Future Prospects at the LHC", CERN-OPEN-2000-311, 2000.

Chapter 2

Network-based DAQ Systems

In this chapter we will point out the need of a new protocol development. In the first section, we review a typical DAQ system and discuss a reason why many systems adopt a network technology. In the second section, the network-based DAQ system is discussed in terms of requirements for the protocol. In the third section, we will point out problems in the present network-based DAQ systems. In the last section, we show that the need of a new protocol development.

2.1 The DAQ systems of high energy physics experiment

We will begin our discussion by reviewing a typical DAQ system. First, we discuss a typical DAQ system and point out that communications between devices play an important role in the DAQ system. Next we point out that the network technology is essential for the DAQ systems.

2.1.1 Typical DAQ system

The task of a DAQ system is to select rare events which are interesting and to build an event from the detectors.

Figure 2.1 shows a typical DAQ system of high energy physics experiments. The system consists of detector subsystems, an event builder, event processors and mass storage devices.

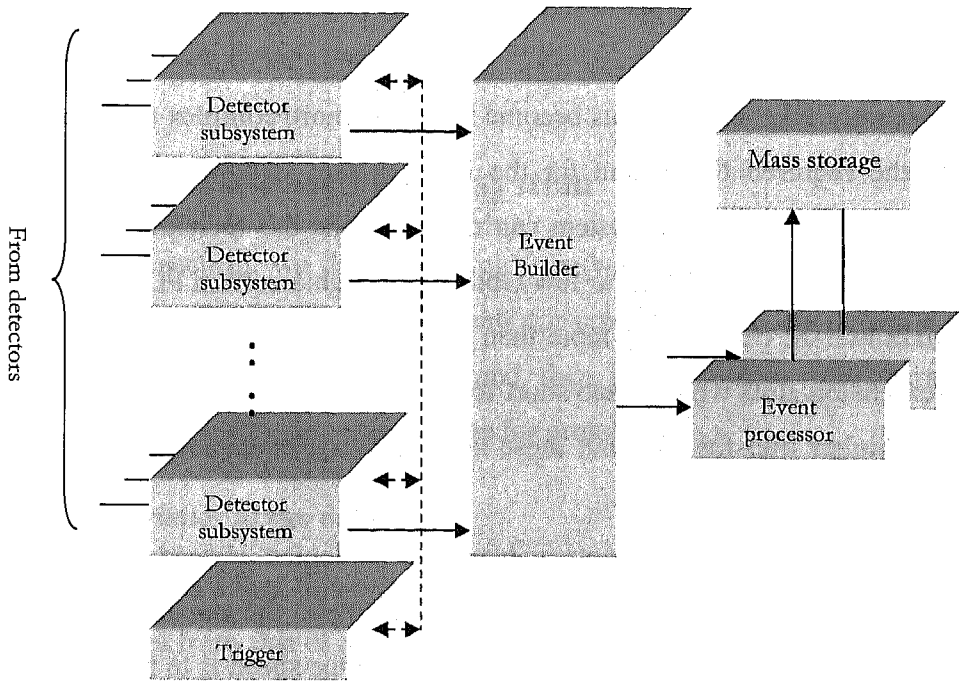

Figure 2.1: Typical DAQ system of a high energy physics experiment.

The detector subsystem consists of front-end devices. The main task of the front-end device is to digitize signals from detectors, being collected together among nearby channels. The pulses from the detectors are first amplified, shaped and discriminated by a front-end circuitry usually mounted directly on the detectors, before being digitized.

Combinations of these signals are sent to a trigger logic, which decides in a short time if the event is interesting and if the current readout agrees with predefined physics requirement.

If the event configuration is to be retained, the detector signals are digitized by front-end devices in the detector subsystem. The event-fragment data for an event are collected, and, then, built for a completion event by the event builder. The event data are sent to event processors which reduce, format, process them and finally record the data on the mass storage.

2.1.2 Detector subsystem

The detector subsystem consists of front-end devices which are organized under a common readout system. The subsystem usually adopts a bus technology for the

purpose. There are standards, for example, CAMAC [2], FASTBUS [3], VME [4], PCI [5], PCI-X [6], and so on.

Recently HEP experiments become more and more complex and sophisticated, and, then, the required throughput for the DAQ system is still increasing from one experiment to another. The subsystem can not adopt the bus-technologies because its maximum data transfer rate is up to few Gbit/s. Although a certain PCI system, i.e. 64-bit 100-MHz PCI-X, can achieve more than a few Gbit/s transfer rates, the system has only two extension slots which is too short for a HEP DAQ system. There exist systems which have higher transfer rates but these have only one extension slot (exist; PCI-X 133, PCI-X 266, PCI-X 533).

Therefore, a system adopts network technologies instead. In last decade, network technologies are rapidly developed. For example, the maximum speed of the Ethernet standard remained at 10 Mbit/s until 1995 but the speed is to be up to 10 Gbit/s in December 2004 and still developing. The speed has been developed a thousand times faster during ten years. Recent network technologies have high performance and one of advantages of network technologies can be found in their high flexibility.

It is note that recently a bus technology that adopts network technologies is appeared, for example a PCI-Express [7].

2.1.3 Event builder

Since the required throughput is increasing, the number of devices which constitute the system is also increasing and communication connections between them become complicated. Since in many DAQ systems the number of event processors is quite smaller than the number of detector subsystems, especially connections in an event builder are to be complicated. The distance between devices in the system is not necessarily short.

In the system, the specialized communication methods have been used because there is not a standard technology whose performance is not enough to process it. However, recent rapid development of network technologies enables to adopt the technologies for an event builder. Since the network technologies have a high flexibility and high performance, the technologies is suitable for an event-builder system and

many DAQ systems adopt network technologies for their event builders, which are called network-based DAQ system.

2.2 The network-based DAQ system

In this section, we discuss a typical network-based DAQ system for a high energy experiment in terms of requirements for protocols. First we define a simplified model of a network-based DAQ system, and, then, discuss a data flow characteristic in the network. Next we discuss requirements for protocols.

2.2.1 Characteristics of the data flows

We discuss a data flow in the DAQ systems. The system has a unique data flow, which is different from that of a general network.

The main task of the DAQ system is to collect a lot of event fragment data of an event and to build the event. Therefore, the direction of the major data flows of communication between detector subsystems and event processors is from the detector subsystem to the event processors.

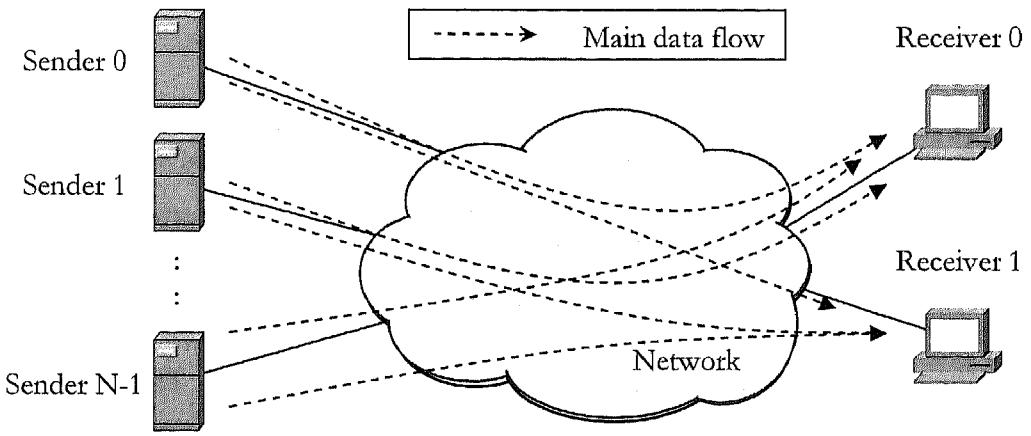

Figure 2.2: Simplified model of network based DAQ system.

Figure 2.2 shows a simplified model of the network-based DAQ system which consists of N senders and two receivers. In the system, the sender represents a detector subsystem and the receiver represents an event processor. Since the receiver collects

event fragment data from many senders, the senders could transmit data to the receiver at the same time.

The characteristic of the data flow is distinguishable different form that in a general network. In general networks, since stations which communicate with each other are decided randomly, there is not a particular direction of these data flows. For example, anyone would get data from a server who we can not know the communication and the direction of data flow before a start of the action. Moreover data is transferred between two stations or one sender to more than one receiver known as multi-cast or broad-cast communication. These communication types are also different from the type in the DAQ systems.

Next we would point out important features of the network in the DAQ systems for designing. Since a general network is usually an open network and connected to Internet in many cases, the number of stations and traffics coming from the outside are dynamically changing. Therefore, it is difficult to predict the network condition and traffic engineering is to be complex. On the other hand, since the network for the DAQ system is usually closed and the communicated partners are decided already, we can design a special network and know network parameters. We can design and construct a high performance system with using its characteristics.

2.2.2 Network protocol

We will discuss network protocols to review concepts in this subsection. The communication between computers over a computer network is governed by a set of rules called a protocol. Its specification consists of two elements: the syntax and semantics. The syntax of the protocol specifies all messages exchanged between the computer, their formats, and the meaning of each field of the message. The semantics of a protocol specifies the actions taken by each computer when specific events, such as arrival of message or timeouts, occur.

The protocols have a layered model and it is usually designed for one of them in the layers. The model is called the Open System Interconnection (OSI) reference model [8] that was developed by the International Organization for Standardization (ISO) as a computer protocol architecture and as a framework for developing protocol standards.

Figure 2.3 shows the OSI model that consists of seven layers: physical, data link, network, transport, session, presentation, and application.

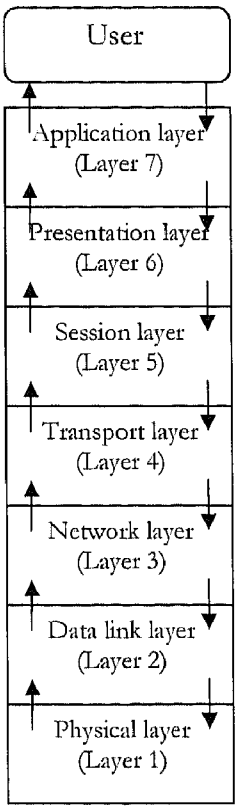

Figure 2.3: OSI model

- **The physical layer (Layer 1):** concerned with transmission of unstructured bit streams over physical medium; deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium.
- **The data link layer (Layer 2):** provides for the reliable transfer of information across the physical link; sends blocks (frames) with the necessary synchronization, error control, and flow control.
- **The network layer (Layer 3):** provides uppers with independence from the data transmission and switching technologies used to connect systems; responsible for establishing, maintaining, and terminating connections.

- **The transport layer (Layer 4):** provides reliable, transparent transfer of data between end points; provides end-to-end error recovery and flow control.
- **The session layer (Layer 5):** provides the control structure for communication between applications; establishes, manages, and terminates connections (sessions) between cooperating applications.
- **The presentation layer (Layer 6):** provides independence to the application processes from differences in data representation (syntax).
- **The application layer (Layer 7):** provides access to the OSI environment for users and also provides distributed information services.

In this thesis, a word of the network is used for the physical and the data link layers; a word of the protocol is used for the network and the upper layers of the OSI model.

2.2.3 Requirements for the protocols

We discussed characteristics of the network in the DAQ system. In this subsection, we discuss a suitable protocol for the characteristics and point out requirements for the protocols.

A network such as Ethernet, which is only to deliver information to a destination station, the upper protocol is responsible for a reliable delivery. Therefore a network is usually used with a protocol that has a mechanism of reliable data delivery.

We discuss requirements for the protocols in the DAQ systems. The network technologies are rapidly developing and it is influenced by commercial trends. Therefore, a protocol should not be affected by the trends. In general, network protocols are independent of network technologies. The data comprises fragmented data from the many detectors for an event and data losses are not allowed in many systems then a reliable data delivery is required. The size of the DAQ system could be large or small then scalability is required. The system should satisfy requirements of an experiment then a quantitative design is required. We summarize above requirements in the following.

- **Independence from the physical and network layers:** The network technologies are rapidly developing and it is influenced by commercial trends.
- **Reliable data delivery:** Data losses are not allowed in many systems because the data comprises fragmented data from the many detectors for an event.
- **Scalability:** The size of the system could be large or small.
- **Quantitative network design:** The system should satisfy requirements of an experiment.

2.3 Problems in present systems

In this section, we point out problems in present DAQ systems. Many present systems adopt standard protocols. In the first subsection, we discuss reasons why they adopt those standards. In the last subsection, we point out problems in the systems.

2.3.1 Present systems

Many of systems have adopted Ethernet for the network, and the TCP/IP protocol suite for protocols.

Ethernet is one of local area network (LAN) technologies and has high performance and various interface types. For the reasons mentioned above, Ethernet is de facto standard interface for PC's, computers and telephones of the IP telephony, and it has been widely used at present. As a consequence, these devices become cost-effective and many DAQ systems have adopted the Ethernet technology.

Next, we discuss protocols in present systems. Recall our earlier discussion about requirements for the protocols. Those are as follows:

- Independence from the physical and network layers
- Reliable data delivery
- Scalability
- Quantitative network design.

Since the TCP/IP protocol suite is developed based on Internet, the protocol suite has an independence from the physical and network layers, strong robustness,

reliability, and scalability. Internet has been grown explosively and the number of hosts connected to Internet networks exceeds 100 million. Since Internet has widely used, the TCP/IP protocol suite has been adopted for a standard protocol suite by many operating system (OS), so we can make up a network program very facilely by using standard functions of the OS such as *socket* functions of Linux. This is not a requirement but it is advantage in developing a system. One requirement have been left, which is a quantitative network design. The requirement can not be satisfied with the TCP/IP protocols suite and it is problems in present system. We will discuss the problems in the next subsection.

2.3.2 Problems

Let us discuss the problems in detail. The problem is that we can not quantitatively design a network of the DAQ system with a standard reliable protocol such as TCP.

First, we discuss a quantitative design. The system should satisfy requirements of an experiment and the specific requirement for a DAQ system is a system throughput. In a network-based DAQ system, that requirement is decided by network performance. There are two parameters which are characterized network performance, as follows:

- Transfer latency
- Throughput.

If we know above two parameters, we can design stations which are connected to the network, for example, transfer latency decides a capacity of buffer in stations and throughput decides a system structure. In order to know these parameters, we should understand traffic behavior in the network.

However, it is difficult to understand the traffic behavior of the standard reliable protocol in the DAQ system. The reason of the difficulty is related to the communication types of the protocols. There are two communication types which are a uni-cast type and a multi-cast type. The uni-cast type is used for a communication between two stations. The multi-cast type is used for between a sender and many receivers. The communication type in the DAQ system is different from either a uni-cast type or a multi-cast type. Since there is no application same as the system in

general networks, we can not find the suitable protocol for the system among the standard reliable protocols.

When we employ a standard reliable protocol, we encounter a problem of packet losses. We will discuss the problem in detail. Since a standard reliable protocol is designed based on a communication between a sender and a receiver, the protocol is not able to control many senders which independently communicate to a receiver. Therefore, the packet loss is induced by a particular data flow of the DAQ system; the data flow quietly different from the traffic in a general network. The data flows are same directions and heading to the same destination, as earlier discussion, and these flows aggregate at packet switching or routing devices, and, then, many packets are forwarded from input ports to a destination port in short time, which cause a packet buffer over-flow, and, then, packets losses occur. A standard reliable protocol detects the packet loss at the receiving side and tries to recover the lost data with re-transmissions. Since the protocol efforts to recover errors which are caused of the data transfer under various network conditions, the protocol needs complicated processes, and, then, the behavior becomes entangled. Therefore the packet loss and its behavior in packet switching devices under aggregating many sources are too complicated to predict and moreover its recovery mechanism makes the behavior prediction to be difficult, which makes a performance prediction of data transfer to be unreliable.

In order to solve the problem, there is a method in general networks. That is called a Quality of Service (QoS) [9]. There are various methods but the main idea is to assign bandwidths of connections. There are two assignment methods; one assigns a fix bandwidth known as constant-bit-rate (CBR), the other one dynamically assign a bandwidth known as variable-bit-rate (VBR). If the transfer rate of senders always constants, the CBR methods solve the problem. But since the data size of event fragment is usually variable and the transfer rate always varies, the CBR method can not solve the problem. On the other hand, the VBR method seems to be good for the DAQ system. The problem is that the VBR method is not able to achieve a high efficiency. Since a protocol of the VBR method controls devices and assigns a bandwidth, the protocol should know status of senders and quickly control the devices for achieving a high efficiency. However, it is difficult to know status and quickly control devices. In the system, since a packet loss can not be permitted, the QoS protocol needs special

bandwidth for it. The QoS method is equipped with additional protocols which require special network devices supporting the protocol. Additionally, the protocol requires complicated processes and a CPU deal with it. Therefore, the performance depends on a way of implementing the protocols [10] [11].

As discussed above, we can not quantitatively design a network of the DAQ system with a standard reliable protocol such as TCP because there is no method for avoiding packet losses at high efficiency data transfers.

Finally we discuss a workloads of protocols. Since a standard reliable protocol such as TCP with a QoS protocol requires a heavy workload [11] [12], we should adopt a high performance device, for example, a high speed CPU, a high speed processor, and so on. Since the process is complicated, the workload is to be a serious problem when a receiver processes the protocols.

2.4 The goal of this study

The standard reliable protocols such as the TCP/IP protocol suite have problems in the DAQ system. There are candidate for a network of the DAQ system, for example Ethernet, but there is no candidate for the protocol. In order to solve the problems, we propose to develop a new protocol for a DAQ system for HEP experiments.

The goal of this study is to develop a new protocol for a DAQ system. The protocol should satisfy requirements which are discussed in section 2.2. And we add one more requirement to achieve a highly transfer efficiency more facilely, which is concerned with a light workload that is discussed in the previous section. The requirements are as follows:

- Independence from the physical and network layers
- Reliable data delivery
- Scalability
- Quantitative network design
- Light workload.

By developing the protocol, we aim to construct a high performance DAQ system.

References

[1] K. Abe *et al.*, "Letter of Intent for KEK Super B Factory", KEK report 2004-04.

[2] IEEE, "IEEE standard Modular Instrumentation and Digital Interface System (CAMAC) (Computer Automated Measurement and Control)", IEEE Std 583-1975.

[3] IEEE, "IEEE standard FASTBUS Modular High-Speed Data Acquisition and Control System", ANSI/IEEE Std 960-1986.

[4] IEEE, "VMEbus Specification", IEC821/IEEE Std 1014.

[5] E. Solari and G. Willse, "PCI and PCI-X Hardware and Software: Architecture and Design", Annabooks, 2001.

[6] T. Shanley, "PCI-X System Architecture", Addison-Wesley, 2001.

[7] R. Budruk, "PCI Express System Architecture", Addison-Wesley, 2003.

[8] ISO, "Information Proceedings Systems - Open Systems Interconnection – Basic Reference Model", International standard 7498, 1983.

[9] G. Armitage, "QUALITY OF SERVICE IN IP NETWORKS", Macmillan Technical Publishing, 2000.

[10] Y. Yasu, et al., "Quality of Service on Gigabit Ethernet for Event Builder", The 3rd International Data Acquisition Workshop on Networked Data Acquisition Systems, Lyon, France, October 20, 2000.

[11] Y. Yasu, et al., "Quality of Service on Linux for the Altas TDAQ Event Building Network", CHEP2001.

[12] R.W. Dobinson *et al.*, "IEEE 802.3 Ethernet, Current Status and Future Prospects at the LHC", CERN-OPEN-2000-311, 2000.

Chapter 3

Data Collection Protocol

We have developed a new protocol that is suitable for the DAQ system of HEP experiment, which satisfies requirements which are discussed in previous chapter. We call the new protocol Data Collection Protocol (DCP).

In this chapter, we discuss DCP in detail. In the first section, we show a design concept. In the second section, we discuss a model that is assumed in designing DCP. Next the key mechanisms are discussed. In the last section of this chapter, we give an overview of our protocol specification.

3.1 The design concept

In the previous chapter, we discussed the problem in the present systems, and pointed out a need of development of a new protocol. The problems come from a particular communication type of the DAQ systems which induces packet losses.

In order to solve the problems, we design a protocol with a mechanism that controls transmitting of senders and avoids packet losses. By avoiding the packet losses, we can quantitatively design a network. If the losses are completely avoided, a reliable data delivery is realized. But actually sometimes packet losses occur in a receiver then we develop a simple mechanism for a reliable data delivery. We design the protocol that the workload is subdivided as a light workload for each station.

Finally we discuss employment of other protocols. We do not employ other protocols because DCP does not require other protocols and we exclude uncertainty of influence from the protocols. We will give an example to show the influence. To take the case of an IP, the protocol needs control protocols which are known as ICMP (Internet control message protocol) [1] and ARP (Address resolution protocol) [1].

Therefore if we employ IP then we should understand these behavior and influences on DCP.

3.1 Assumed configuration

In this section the assumed configuration in designing DCP is discussed. The configuration is a base of our design. The configuration includes a network entity, a network structure and requirements for network devices. DCP is specialized for DAQ systems and we do not consider that the protocol used in general networks such as Internet.

3.1.1 Entity

We assumed that the DCP was used for collecting data via a network. We define network entities and terminologies.

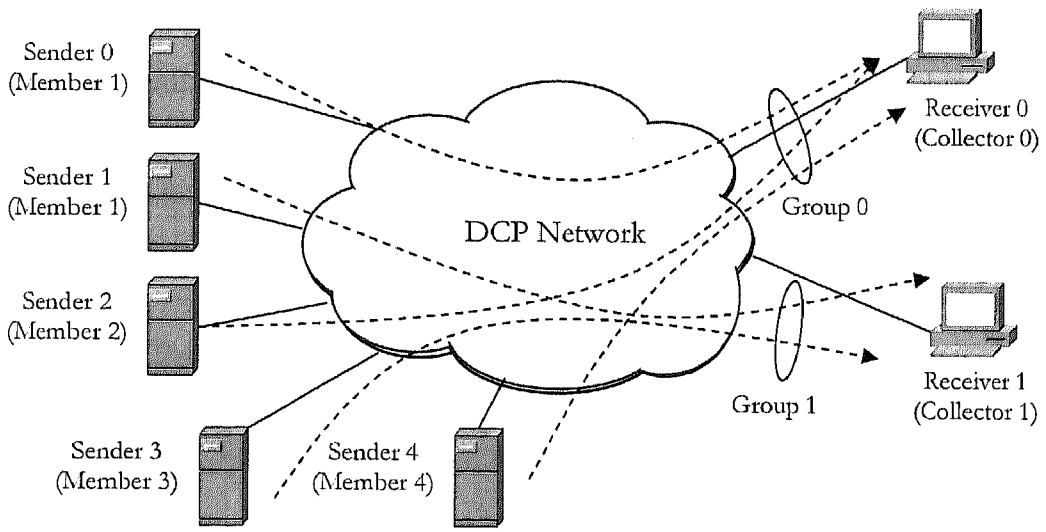

Figure 3.1: DCP basic operation model

Figure 3.1 shows a basic configuration of the DCP network system. There are two groups; one is group 0 that consists of sender 0, sender 2, sender 4 and receiver 0; the other is group 1 that consists of sender 1, sender 3 and receiver 1. In the group 0, three senders transmit data to the receiver and the data transfer potentially aggregates at the receiver via the network. In group 1, the data transfer aggregates as similar as the

group 0. Senders are something like front-end devices and receivers are something like event processors in the DAQ system. In a DCP network, the sender is the member of the group and the receiver is the collector of the group. In a similar fashion as TCP/IP adopts a client-server model, DCP defines a member-collector model.

A member and a collector must belong to a group but it should not belong to two or more groups. If a station uses groups, it should have two or more members or collectors.

3.1.2 Network

We design DCP that is independent of a physical and data link layers. Since we should choose a network technology for an implementation, we adopt Ethernet. The reasons of adopting Ethernet are as follows:

- High performance
- Simple mechanism
- Cost effective.

In this subsection, we would like to define a configuration and terminology of a DCP network. Since we have adopted Ethernet for constructing the network, the network devices fall into a category of Ethernet network devices. Since a multi-group DCP network system is more complex than a single-group system, we limit the discussion to the single group system. The multi-group system will be discussed in chapter 6.

Figure 3.2 shows a basic model of a single-group DCP Network system. In this figure, the HUB designates an Ethernet switching hub (HUB), where the word of HUB is used for a device that has only packet switching function. There is an Ethernet intelligent fast switch that has QoS functions or various algorisms of schedulers of packet queuing, which can be readily used. But DCP needs only packet switching function and others are not required. As the figure, the network is very simple and need not Layer 3 network switches or routers. Finally we note that a network topology of Ethernet must logically be a spanning tree topology.

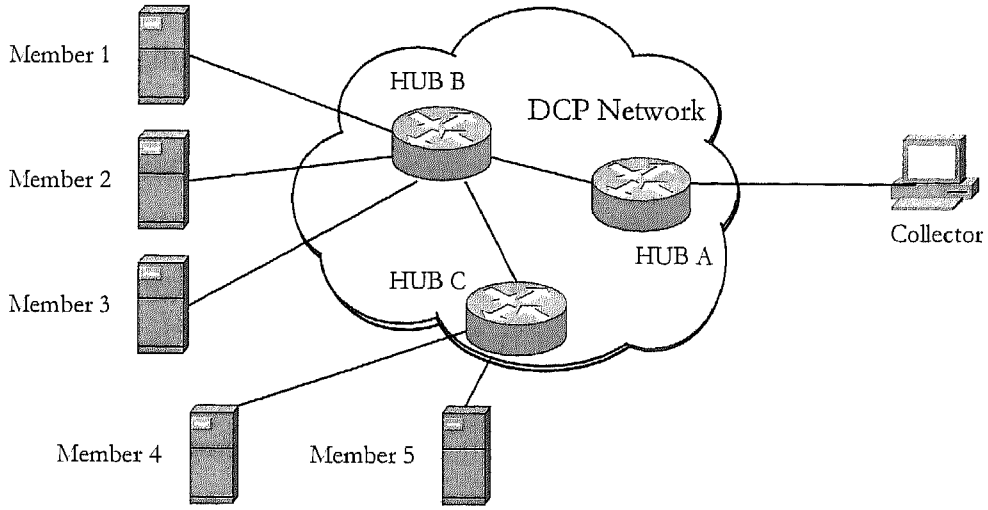

Figure 3.2: Single Group DCP Network system

3.1.3 Switch

We would like to define HUB specifications. HUB is a frame forwarding device to forward received frames to a destination port according to the MAC address.

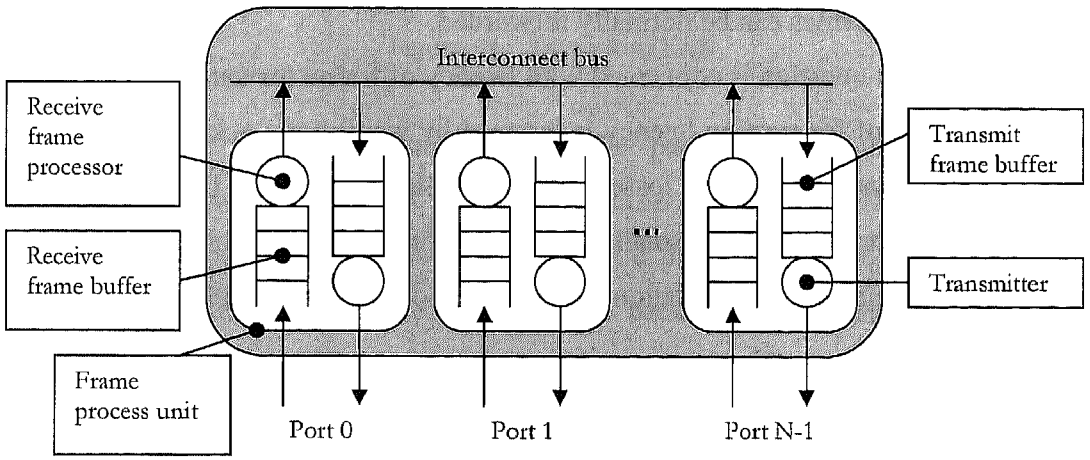

Figure 3.3: Model of a HUB

Figure 3.3 shows a model of a HUB that consists of N frame process units and an interconnect bus. The HUB is assumed to utilize a store-and-forward mechanism, where frames are completely buffered at the input port before being sent to the output port via the interconnect bus. We assume that a capacity of the transmit frame buffer

and the receive frame buffer are compatible for one frame. The receive frame processor and the transmitter have a frame buffer where the frame is processed after some waiting period.

First a frame is received and put into the receive frame buffer. Second if the receive frame processor is idle, that means there is no frame in the processor, the frame is transferred into the processor and checked by a frame-check sequence provided in the frame (Appendix A); when the processor is busy, that means there is a frame in the processor, the frame waits for turning idle. Third the receive frame processor requests a send port for forwarding frame. If the transmit frame buffer is empty and there is no request from other port, the acknowledgement of the request is replied soon and the processor sends the frame to the transmit frame buffer. But in other case the processor waits for the acknowledgement. When a competition with other port is occurred, the priority for a forwarding frame is decided by a round-robin algorithm. Finally the frame is transmitted to the line by the transmitter.

In this model, frame losses occur when more than one port forward frames to a single port simultaneously, and, then, the frame is discarded due to buffer overflow. In the other words, if the port that forwards a frame to another port is unique for a given time interval, the frames can be normally forwarded.

The next question is concerned with latency of frame forwarding. In this model, a large variation of forwarding latency is induced by waiting the acknowledgement. As same as the discussion of frame losses, if the port that forwards a frame to the port is always unique, the variation is small because the processor can forward to send port without waiting-time and the variation is decided by only frame length.

3.3 Key mechanisms

In this section, we discuss key mechanisms of DCP. In the first subsection, we discuss a sender control mechanism that is a main mechanism of DCP. In the next subsection, a data transfer mechanism is discussed.

3.3.1 Sender access control

We design DCP that avoids packet losses with a token passing mechanism. The mechanism has been adopted for an access control of senders by some LAN's, for

example, a Token-Ring [2], and so on. As a beginning, we will introduce the mechanism with the historical overview. Next, we discuss the mechanism in DCP.

We review a Token-Ring network for explaining a token passing mechanism. The Token-Ring has adopted the mechanism for an access control of senders. The Token-Ring was invented at IBM research labs and was based on a ring topology as shown Figure 3.4.

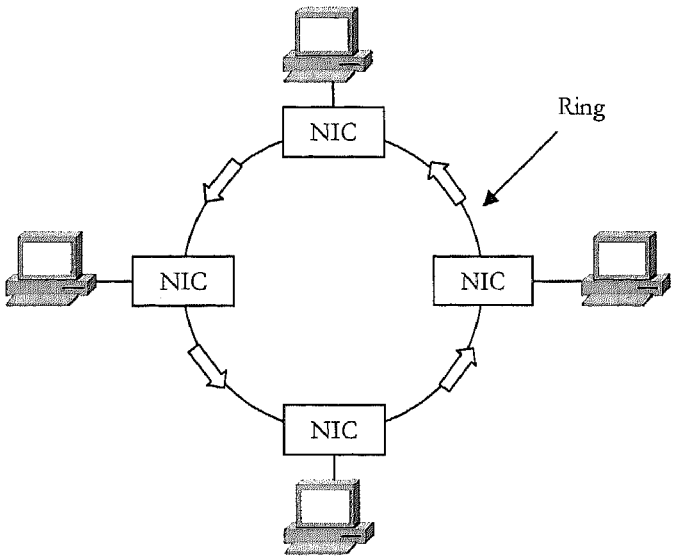

Figure 3.4: Token-Ring LAN

A sender inserts data into the ring. The packet goes around the ring and is copied by the Network Interface Card (NIC) specified in the destination address field of the packet. The packet continues its flow around the ring back to the sender which removes the packet and compares the received packet with the packet sent for error control. If two or more NICs attempt to transmit simultaneously, interference would occur. In a Token-ring LAN, access to the ring is controlled by a token, a special bit pattern that circulates through the ring. The only NIC that has the token can transmit data. When the transmitting is done, the NIC passes the token to the next NIC in the ring. If a NIC does not have any packets to transmit, the NIC just passes the token to the next station. The Token-Ring is superior to other LAN technologies in terms of an access control with penalty of a low flexibility.

Next, we discuss the mechanism in the DCP network. DCP establish a token-ring logically on a network so that it keeps independence from a physical layer and a

network layer. Figure 3.5 shows that a token is passed between members. Since there is always only one sender that is transmitting data to a receiver, a packet loss does not occur in a network. Moreover, since the mechanism makes fair chances to transfer data from members to a collector, network resources are dynamically assigned.

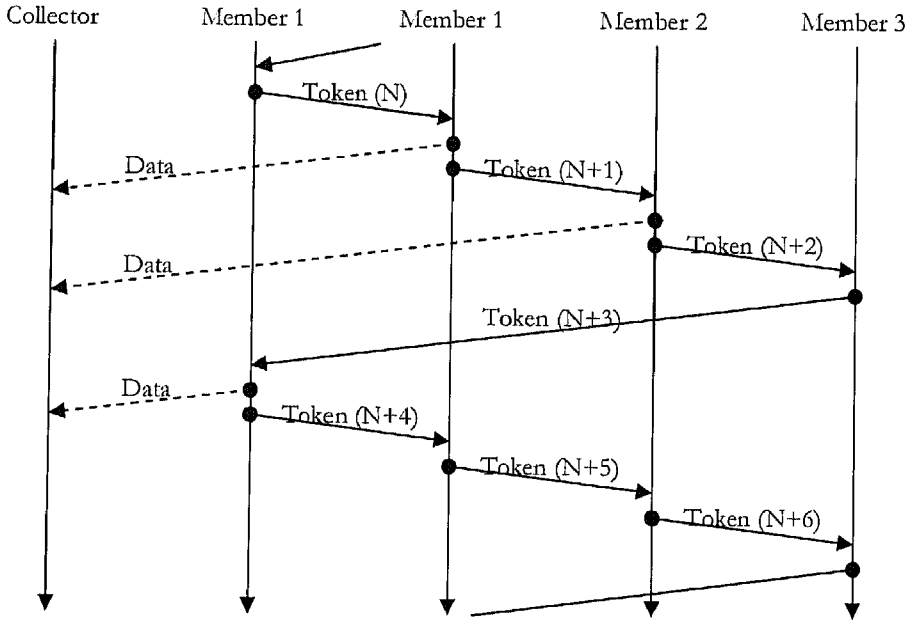

Figure 3.5: Token passing mechanism in DCP Network

In the figure, the number in the parentheses displays a token-number that is used for checking duplicate and wrong token-packets. If a member receives a token packet that has an unexpected token-number, the member discards the packet. In the network, a token packet is normally not lost, however if the packet losses occurs, the packet is recovered by a re-transmission mechanism by the member. The mechanism works by a timeout mechanism that counts time interval between received token packets. If a token does not arrive at a member within the time interval, the member requests a re-transmission of the token for the previous member in the logical token-ring with a packet that has the expected token-number. The member which invokes the retransmission procedure is uniquely identified by the token-number.

By the token-passing mechanism there is always only one member that is transmitting data to a collector, the member tries to use a full network resource up to the limit of a physical interface. Since a period during the member uses a line for

transmitting data depends on the data length, if the member has so long data, the period should be so long. Eventually this long period brings to the system instable. In order to avoid the instability we introduce a maximum length to transmit data, which is called limited service.

3.3.1 Data transfer

Our protocol causes no congestion in the principle. Nevertheless, a packet loss rarely occurs in the network. If the loss occurs, there are two main reasons. One comes from a bad packet that has a bit error and the other comes from a buffer overflow of a collector. The bit error occurs by a bad network condition (e.g. electric noise etc.). The buffer overflow occurs by an over load of the collector. If that is able to receive up to the rate of the interface at anytime, a packet lose rarely occurs. However, since a collector usually processes other background tasks, the collector is not able to receive all packets in same circumstances, and, then, a packet loss occurs. Therefore, a mechanism of a reliable data delivery is required and implemented.

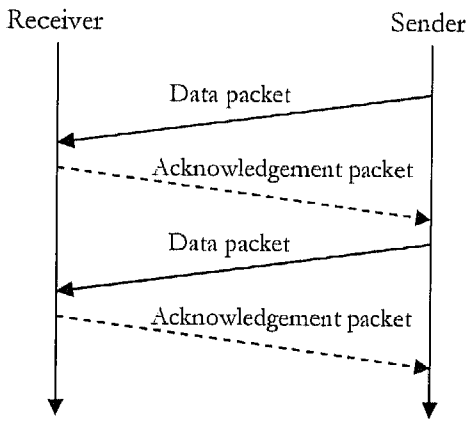

Figure 3.6: Basic data-acknowledgement control method

In order to reliably transfer data, we employ a data-acknowledgement control method. Figure 3.6 shows a basic data-acknowledgement control method. First, the sender transmits a data packet to the receiver. Second, when the receiver receives the packet, it transmits an acknowledgement packet to the sender. In this while, the sender is waiting for the acknowledgement packet and holds transmitted data. Third, the sender receives the packet and transmits the next data. If the sender does not receive the acknowledgement packet within a time interval that is measured by a timeout

mechanism, sender re-transmits the data. In this way, a reliable data delivery is realized with penalty of low transfer efficiency due to waiting-times for data- acknowledgement packets.

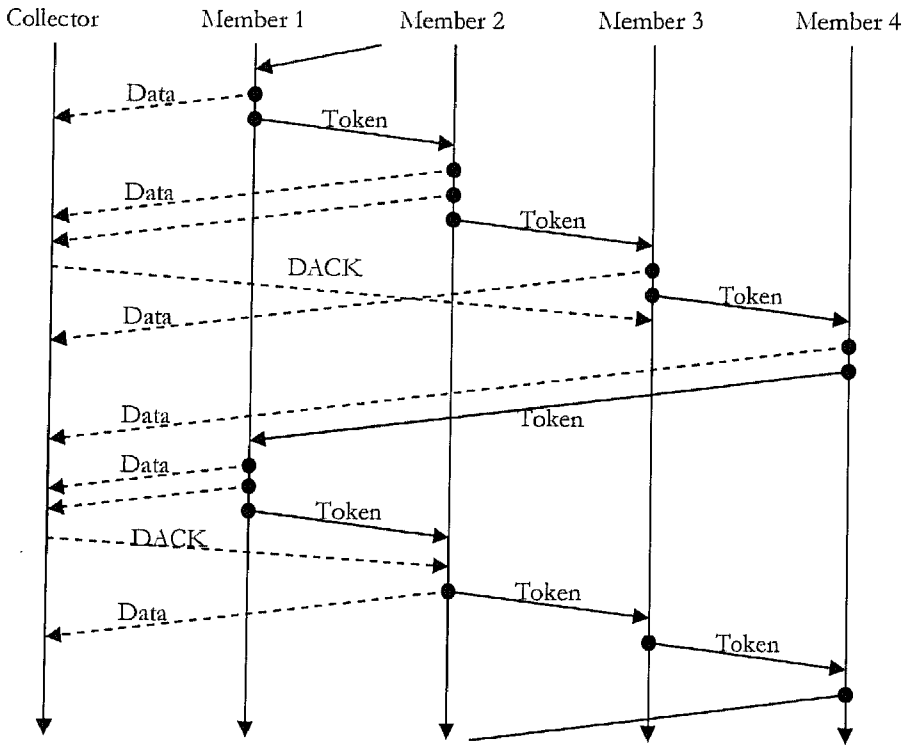

Figure 3.7: Sliding window mechanism in a DCP network.

In order to overcome the disadvantage related with the low transfer efficiency, we adopt a sliding window mechanism. Figure 3.7 shows the mechanism in the DCP network. In the sliding window mechanism, DCP use a sequence number (SN) and an acknowledgment (DACK). Artificially, each octet of the DCP data is logically assigned a SN. Senders transmit data to the receiver with the start SN and the data length. This mechanism enables a sender to transmit multiple bytes or packets before waiting for an acknowledgment. The window that is employed for the sliding window mechanism is the number of data bytes that the sender is allowed to send before waiting for an acknowledgment. Initial window sizes are indicated at the time of initial setup.

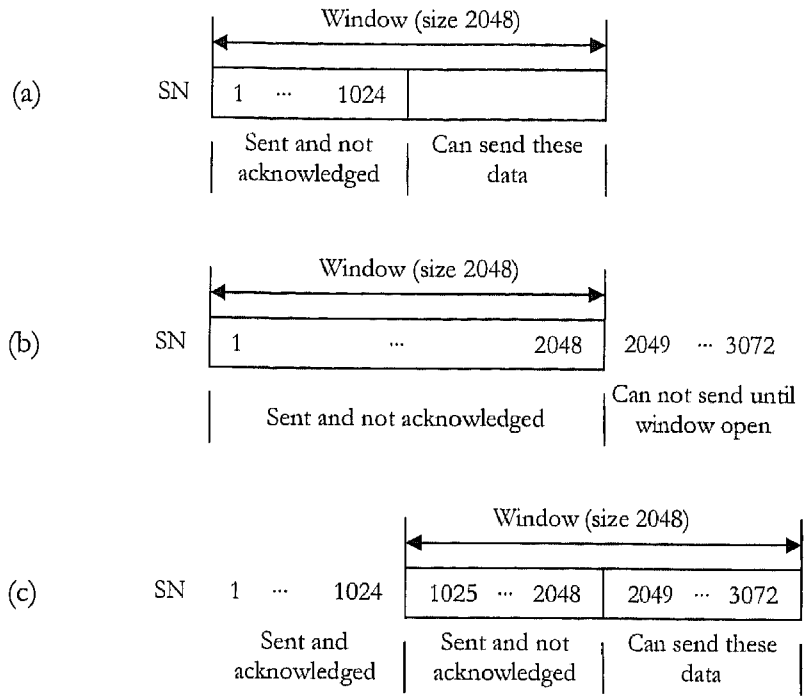

Figure 3.8: Sliding window mechanism.

Figure 3.8 illustrates the evolution of the sliding window. Each number represents a DCP data SN. In the figure, the current window size is assumed to be 2048. In Figure 3.8 a, the member might have a SN to send (numbered 1 to 1024) to a collector and then would place a window around the first 2048 bytes and transmits them. In Figure 3.8 b, the member might have more data to send (numbered 1025 to 3072) to a collector but an acknowledgment has not been received from the collector. Therefore the member transmits first 1024 byte (SN 1025 to 2048). It would then wait for an acknowledgment. In Figure 3.8 c, the receiver would respond with an ACK has SN=1025, indicating that it has received bytes 1 to 1024 and is expecting byte 1025 next. The member then would move the sliding window 1024 bytes to the right and transmit bytes 2049 to 3072. The collector would respond with an ACK has SN=3073, indicating that it is expecting SN 3073 next. We note that actually the data is fragmented into packets and the size is decided by the data link layer protocol, for example that of Ethernet where the size is 1484 bytes. This mechanism has a reliable data delivery as well as a data flow control between a sender and a receiver.

Since we employed the data-acknowledgement control method, DCP has a re-transmission mechanism. The collector compares the start SN and the expected SN

every data received. If these values are equal, the collector sends an ACK packet to the sender. If these values are not equal, the collector sends the sender a re-transmission request packet (RETRANS) that has expected start SN and the member then re-transmits data with the requested start SN to the collector. The DCP data are reliably transferred in a high efficiency by means of the sliding window mechanism.

3.4 Overview of the functional specification

In this section, we give an overview of the DCP functional specification.

3.4.1 Frame structure

At this time, we adopt Ethernet for an implementation and discuss DCP on Ethernet. The DCP data is fragmented and encapsulated in an Ethernet frames. Figure 3.9 shows the Ethernet frame structure. The MAC header consists of a preamble, a start frame delimiter, address fields and a length/type field. The payload carries user data, a DCP packet is encapsulated in this payload, and FCS (frame check sequence) is used for error detection in an Ethernet frame.

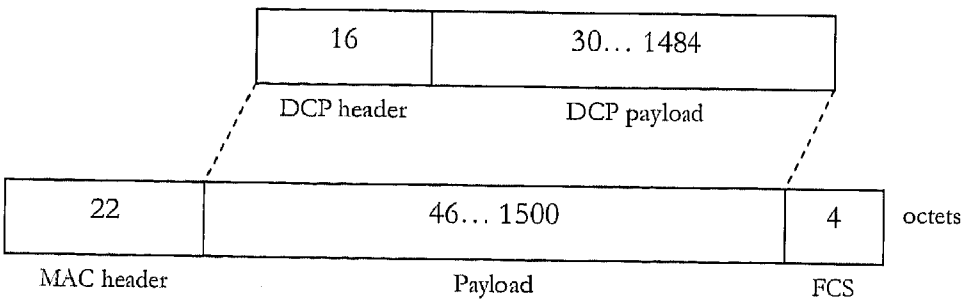

Figure 3.9: DCP frame structure

A DCP header length is 16 bytes and consists of a packet type and the DCP parameters. The meaning of the parameters is depends on the packet type, which detail will be discussed the header in the next subsection. A DCP payload carries user data, for example, fragment event data. If the packet carries no data or the DCP payload length is less than 30 bytes, the payload field is padded with invalid data and the length is to be 30 bytes.

3.4.2 Header format

A DCP header length is 16 bytes, which format is illustrated in Figure 3.10. The transmitting order is from bit 0 to 31 and top to bottom.

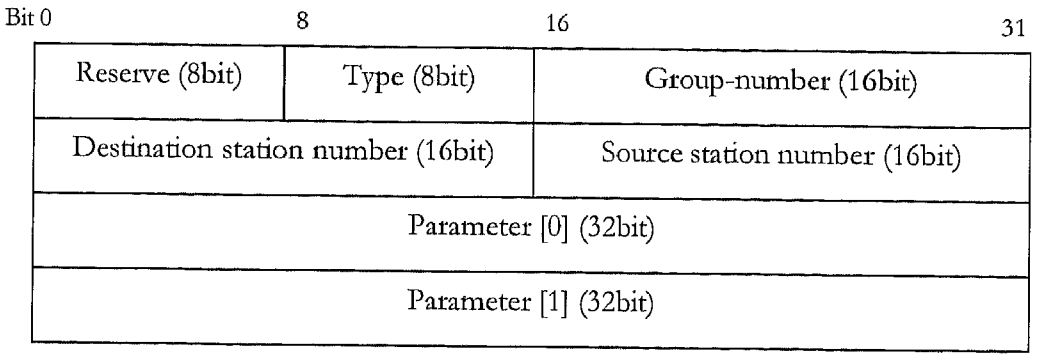

Fig. 3.10: DCP header format

The DCP header consists of seven fields; a reserve-field: a type-field, a group-number field, a destination-station number field and a source-station number field.

- **Reserve:** This field is not used and reserved for a version number and fundamental information.
- **Type:** Indicates a type of the packet, for example, since a token packet has the assigned value in this field, members can identify the packet.
- **Group-number:** Indicates a group-number that is used for identifying a belonged DCP group.
- **Destination-station number:** Indicates the destination station-number of the packet. The station-number is assigned by a collector and the number is unique and the number of a collector is always zero.
- **Source-station number:** Indicates the source station-number of the packet.
- **Parameter:** The meaning of the parameter-fields is depends on the type of packet, for example, a token packet carries a token-number with this field.

3.4.3 Operation

The protocol transfers data with a token passing mechanism that is independent from a network. Therefore, we should establish a logical token-ring on the network. When we want to stop to transfer, we close the ring and free members and collectors from the ring. In this subsection we describe an overview of a sequence of these actions and do not describe details such as error recoveries.

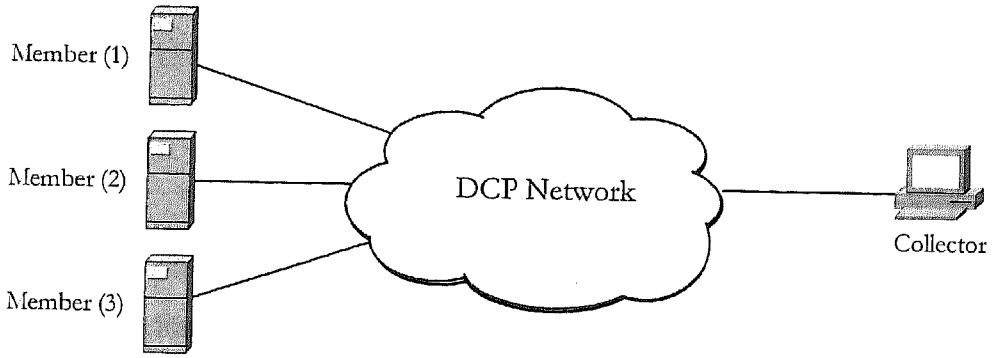

Fig. 3.11: The DCP system.

We will show the DCP action with an example. The DCP system under consideration is shown in Figure 3.11. There are three members and a collector. We would show an outline of the sequence.

1. Make a member list and a sequence of a token passing.
2. Set the DCP parameters to members.
3. Establish a logical token-ring.
4. Transfer data.
5. Close a logical token-ring.

We discuss the sequence step by step. In the beginning, we assume that all members and the collector have no task.

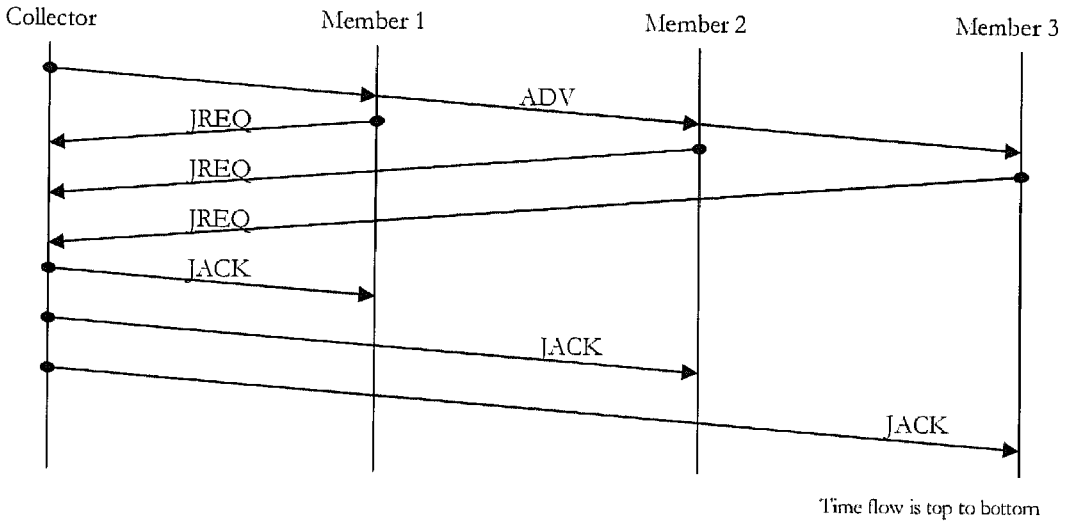

Fig. 3.12: Member list making.

At the beginning of a token ring establishment, a member list is made by the collector. The list is a sequence of a token-passing. There are two methods: one is static method and a list that is prepared by a user: the other one is dynamic method and a list is made by a collector that is looking for members which want to join to a token-ring. Since the former have the list already, we here describe the latter method, which behavior is shown in Figure 3.12.

Since all members and the collector have no task, the collector is waiting for a start request of data transfers by a user and members are waiting for a request packet to join the logical token-ring from the collector, which is called an advertisement. When a user requests the collector to start to transfer data, the collector sends advertisement packets (ADV) with a broad-cast, which request members to join the logical token-ring. If the packet has the group-number that is assigned for a member, the member requests to join the ring with the packet (JREQ). The collector makes the list with those requests and sends confirmed packets (JACK) to the members.

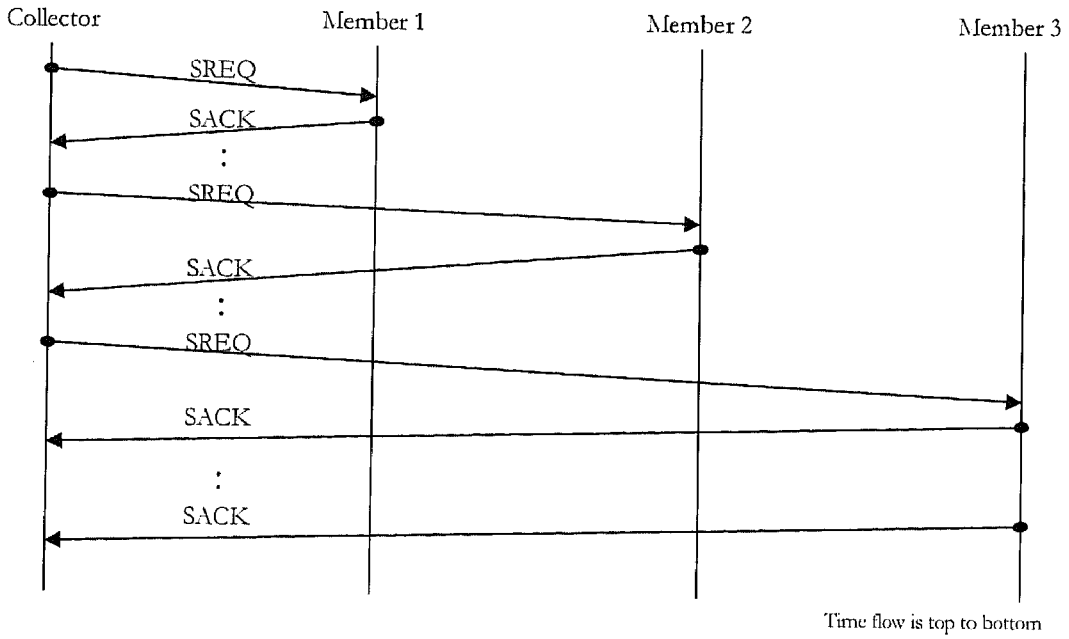

Fig. 3.13: Setting DCP parameters.

Next the collector sets the DCP parameters to members which are joined in the ring which behavior is shown in Figure 3.13. The DCP parameters, for examples, a window-size, data-link layer addresses of the previous and next stations in the token-ring, and so on. The parameters are set by the collector with the packet (SREQ) that has a type of a parameter and value. The member receives the packet and finishes to set the parameter. The member sends an acknowledgement packet (SACK) to the collector for conformation. The collector sets parameters one by one from one member to another. By this way, the preparation for the token-ring establishment has been finished. We call this state a configuration state.

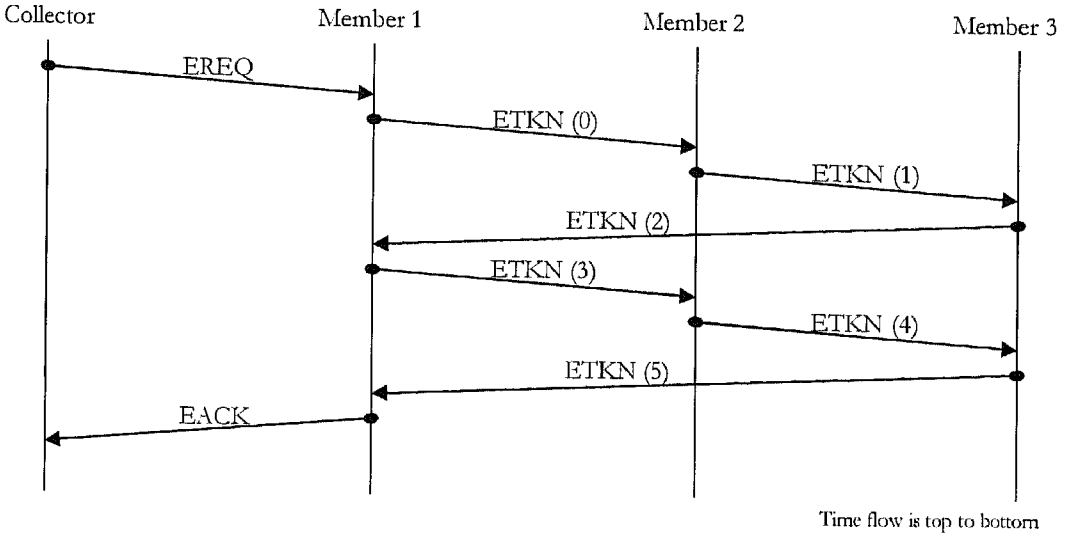

Figure 3.14: Establishment of a token-ring

Next the token-ring will be established and this behavior is shown in Figure 3.14. A primary member is selected among the list by the collector and the collector requests to establish the token-ring with an establishment-request packet (EREQ). When the primary member received the packet, it tries to establish the ring with an establishment-token packet (ETKN). The token packet has a token-number that is used for checking a token which is expected; in the figure, the number is displayed in parenthesis with an ETKN. A member receives the token at first time and registers the token-number and forwards the ETKN with an incremented token number to the next member. Each member calculates the token-number that is expected at next time by adding the registered number and a token-ring-length. The token-ring-length is the number of members, which is set in the configuration state. A member receives ETKN at the second time, it checks the received token-number equal the expectation value or not. If the token-number is expected, the token packet is forwarded but the number is not expected, the token packet is discarded.

At the second time when the primary member receives a correct ETKN, the primary member sends an established-acknowledgement packet (EACK) to the collector and does not forward the token packet anymore. The preparation for data transfers has been finished.

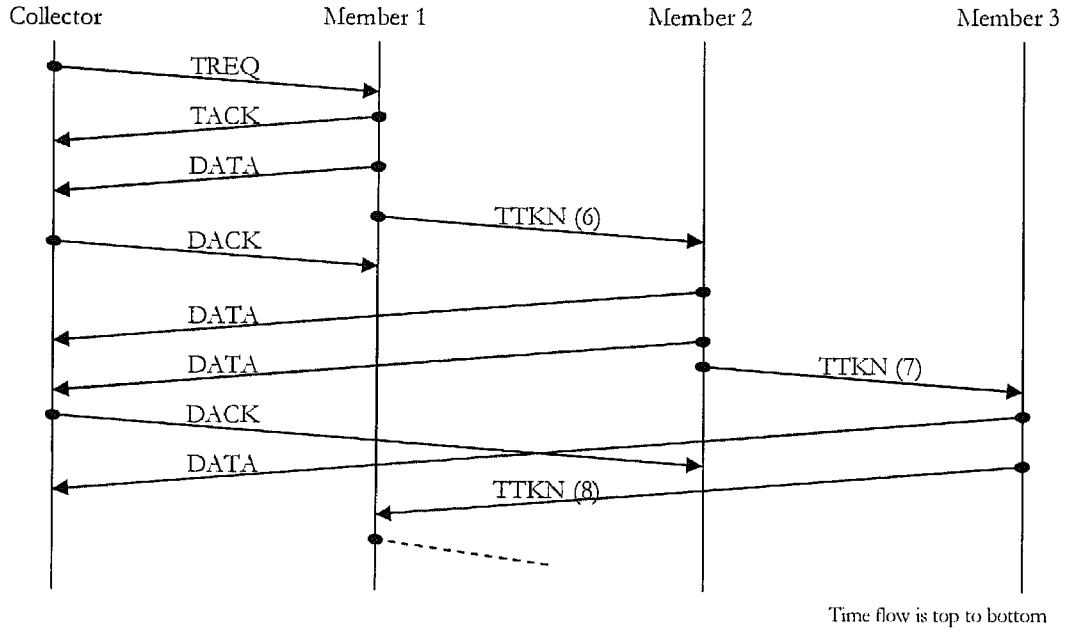

Fig. 3.15: Data transfers

Next the data will be transferred and this behavior is shown in Figure 3.15. The data transfers are started by the collector with a transfer-start-request packet (TREQ). The collector sends the packet to the primary member. When primary member receives the packet, replies to the collector with a transfer-start-acknowledgement packet (TACK) and sends a token packet (TTKN) to the next member, then the token passing is started. The token packet is checked same as the ETKN.

The data are transferred with the sliding-window and re-transmission mechanism which are discussed in earlier section. If a member has data to transmit, the member sends data packets (DATA) to the collector when the member has the token packet. When the collector receives the data packets, it sends a data-acknowledge packet (DACK) to the member. These transfers are continued until to receive a request to stop transfers.

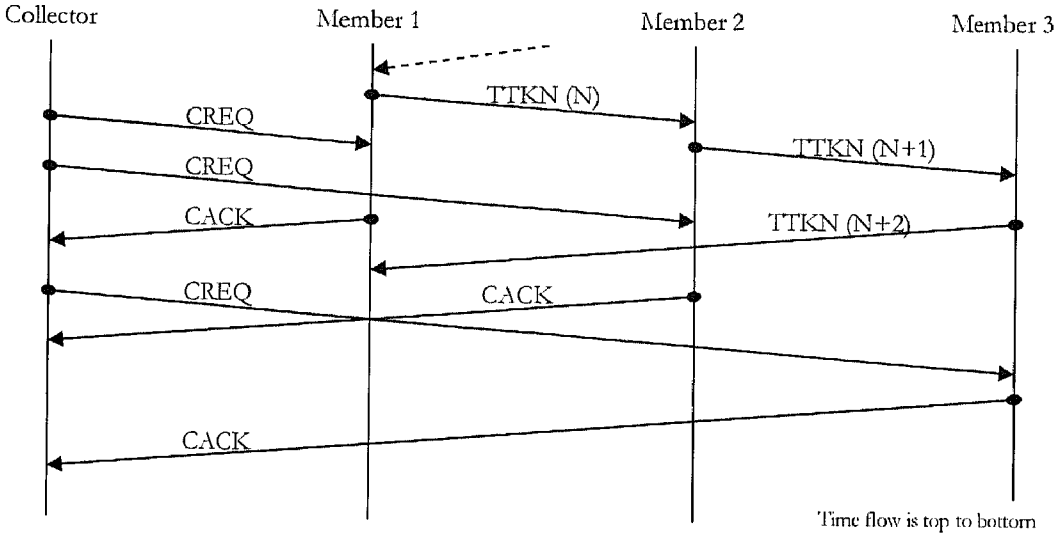

Fig. 3.16: Close a token-ring.

When the collector wants to finish transfer, the collector sends a close-request packet (CREQ) to the members, which is shown in Figure 3.16. When the member receives the packet, it replies with close-acknowledgement packet (CACK). After the collector has confirmed all members are closed, the collector leaves this task.

References

[1] W. R. Stevens, "TCP/IP Illustrated, Volume 1: The protocols", Addison Weslay, 1994.

[2] IEEE, "Token Ring Access Method and Physical Layer Specification", IEEE standard 802.5, 1995.

Chapter 4

Implementation

In this chapter, an implementation of DCP is discussed. Since we design the protocol that has a light workload, we can implement a DCP function on a small hardware device. In order to evaluate the performance and to show the light workload, we have implemented members on a Field Programable Gate Array (FPGA).

4.1 System

Since many of general Ethernet controller chips have a PCI-bus interface, we choose to construct the hardware prototype on PCI-bus of a PC mother board with commodity products.

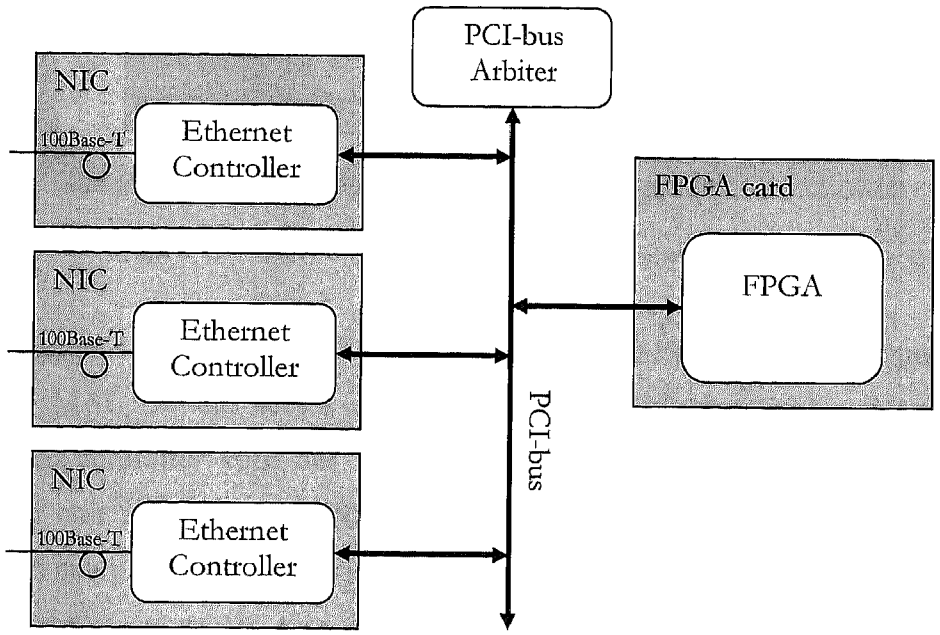

Fig. 4.1: Block diagram of the prototype system

Figure 4.1 shows a block diagram of the prototype system that consists of a PCI-bus arbiter, an FPGA card and three standard NIC's.

The PCI bus is 32-bit 33-MHz type. All cards in the system have PCI-master function and communicate via the PCI-bus each other. These accesses are arbitrated by an arbiter with a round-robin fashion.

4.2 NIC

Figure 4.2 shows a block diagram of the NIC and Figure 4.3 shows a photograph of the NIC. We can found an Ethernet controller, a transformer package and a RJ45 connector. The Ethernet controller is REALTEK RTL8139D [1] which controls to access to a network. The transformer is used for electrical isolation from the network. The card has two interfaces which are a PCI-bus and an Ethernet interface (100BAST-T), which is controlled by the FPGA card to transfer a packet data via the PCI-bus.

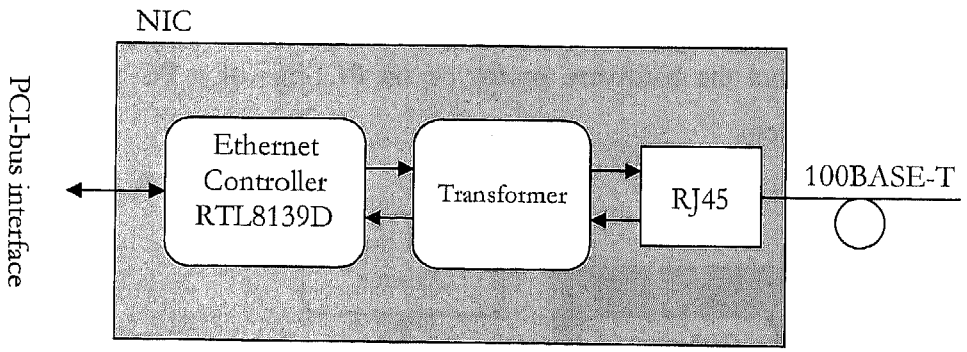

Figure 4.2: Block diagram of the NIC.

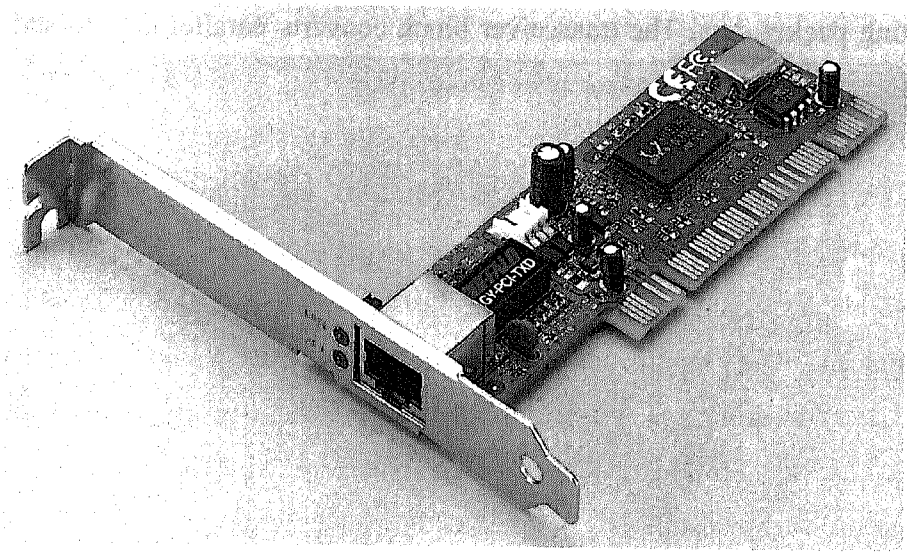

Figure 4.3: Photograph of the NIC.

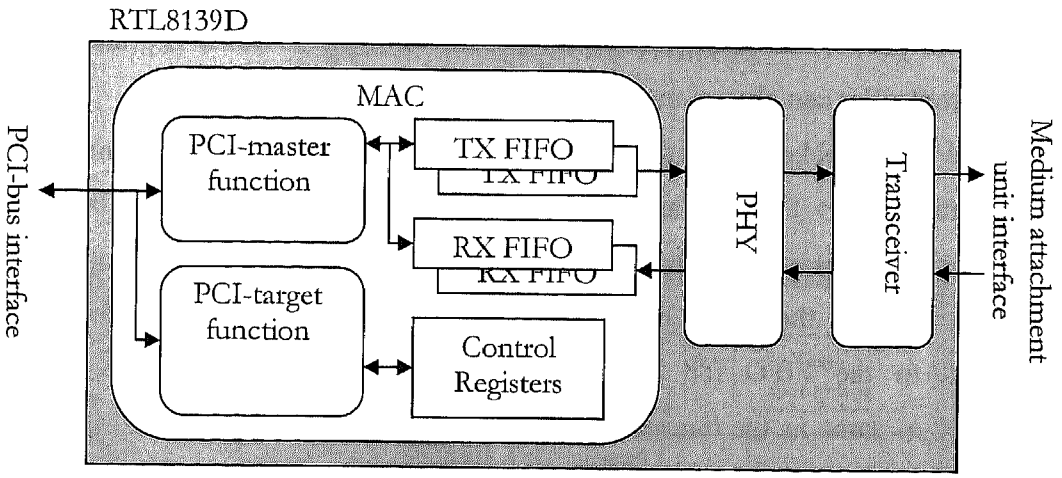

Figure 4.5: Block diagram of the Ethernet controller RTL8139D.

Figure 4.5 shows a block diagram of the RTL8139D. There are three blocks which are MAC (Media access control), PHY (Physical signaling) and transceiver blocks. The controller chip has two main interfaces to a PCI-bus and a medium attachment unit interface. The PCI-bus interface is used for communications to the FPGA card that processes a receiving or a transmitting packets. The medium attachment unit interface is connected to a magnetic component. The MAC block manages to transmit and receive packets. The PHY block codes a transmitting packet and de-codes

a receiving packet data. The transceiver block converts parallel data to serial data for transmitting and serial data to parallel data for receiving.

We explain behavior of the controller in transmitting and receiving packets. When a packet is transmitted by the FPGA card, the card transfers the packet data to a memory in the card and requests to transmit for the controller via a control register in the MAC block. Then the Ethernet controller requests to use the PCI-bus for the PCI-bus arbiter and waits for the acknowledgement. When the acknowledgement signal is received, the controller transfer the packet data from the memory to a transmit First-In-First-Out memory (FIFO) in the MAC block. There are two FIFO for transmitting in the block and the each capacity is 2 Kbytes. If the packet length is less than 2 Kbytes, the FIFO is occupied by the packet. Next the MAC block transfers the packet data to the PHY block. The packet data are coded in the PHY block and transferred to the transceiver block. Finally the packet data are converted to serial data and then transmitted to a network by the transceiver. In the prototype system, we do not adopt a flow-control packet that is defined by the specification IEEE 802.3 [2], and, then, the Ethernet controller can always transmit a packet.

When a packet is being received, the coded packet data come from a network into the transceiver. The data are decoded and converted to parallel data by the PHY block. Next the data are transferred to the MAC block which assembles a packet from the data and check the FCS. Only when a packet has a corrected FCS, the packet is transferred to the FIFO for receiving. There are two FIFO's whose usage are constrained as same as the transmitter FIFO. In order to transfer the packet data in the FIFO to the FPGA card, the controller requests to use the bus for the arbiter and waits for the acknowledgement. When the acknowledgement signal is received, the controller transfers the packet data from the FIFO to a memory in the card.

4.3 FPGA card

Figure 4.6 shows a photograph of the FPGA card that consists of an FPGA, ALTERA EP1S10780C7ES [1]. Figure 4.7 shows a block diagram of the FPGA. The FPGA has one interface to the PCI-bus that is used for communication to three Ethernet controllers on the NICs.

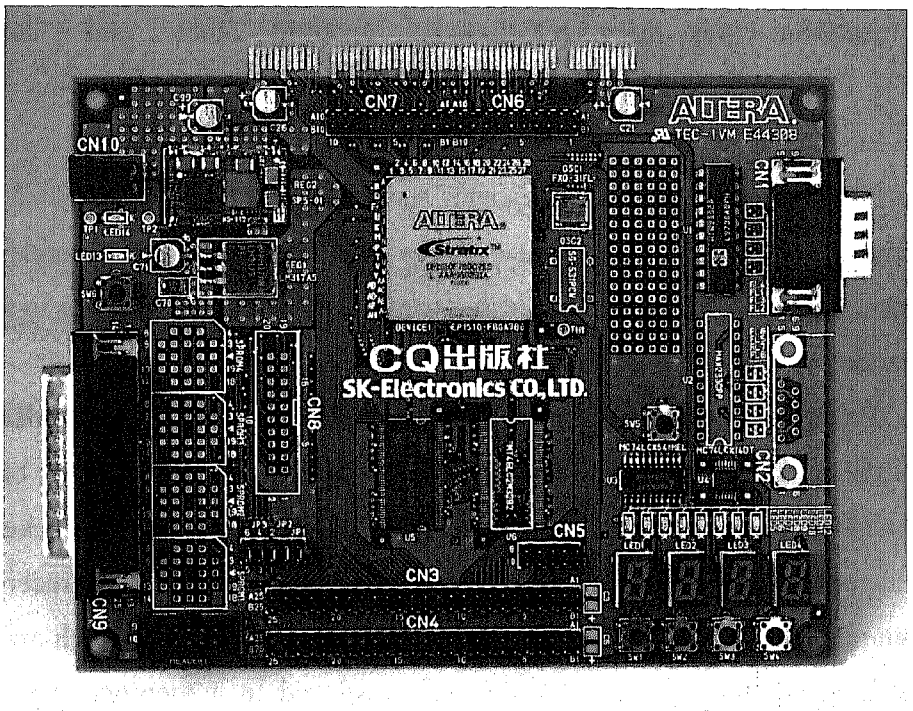

Figure 4.6: Photograph of the FPGA card.

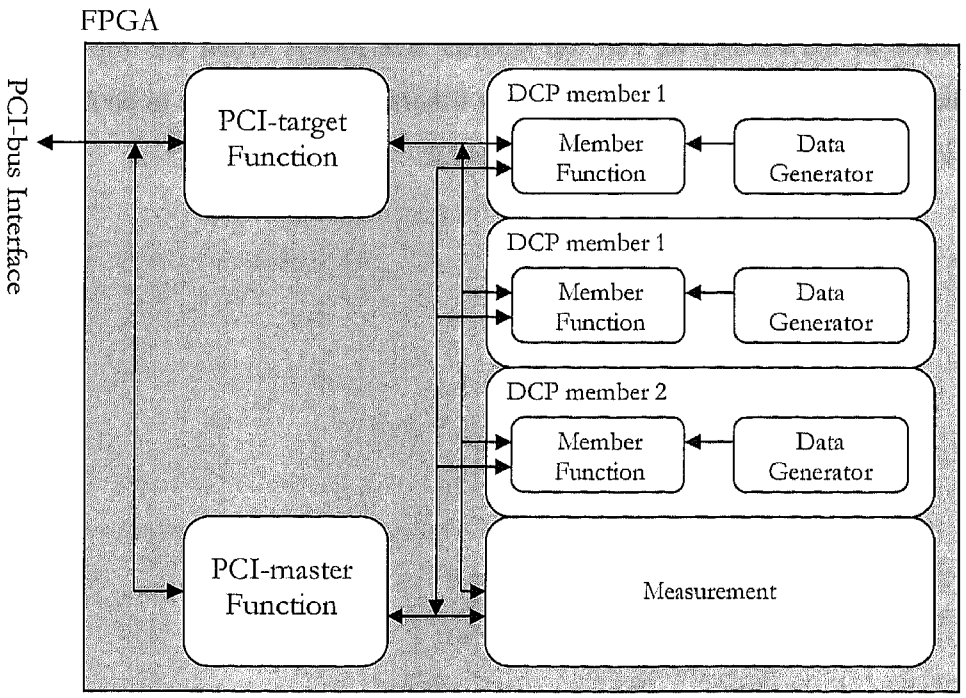

Figure 4.7: Block diagram of the FPGA

The FPGA consists of four functional blocks: a PCI target functional block, a PCI master functional block, three DCP member functional blocks, and a measurement block. The PCI-target block is used to transfer a transmitting and receiving packet data between Ethernet controllers. The packet data are transferred by an Ethernet controller, then the FPGA need to inform the transfer status and the PCI-master block works for the informing. The DCP member block processes DCP packets and generates test data. The measurement block gets and processes parameters of a DCP member to analyze the performance.

We explain behavior of this card. If a packet is received, an Ethernet controller transfers the packet data from a FIFO in the controller to a DCP member block in the card. The block analyzes the packet and decides the next process. If the packet is a DCP packet, the packet is processed, otherwise, the packet is discarded. If the received packet is a token-packet, the block generates DCP data packets from data which are generated by the data generator. The packet data are transferred to a memory in this member function sub-block. Next the block requests an Ethernet controller to transfer the packet data via the PCI-master block. Finally the controller transfers the data from the memory to a FIFO in the controller and then the packet is transmitted.

Since the percentage of used logic elements (LE) for a member is 25% of available LE's of the FPGA, we can use smaller size FPGA when we implement one sender. The DCP implementation for a Gigabit Ethernet is considered to be not difficult because FPGA can run over 100 MHz system clock.

References

[1] REALTEK Semiconductor Corp., "REALTEK SINGLE CHIP FAST ETHERNET CONTROLLER WITH POWER MANAGEMENT AND MULTI-FUNCTION RTL8139D (L)", 2001.

[2] IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE standard 802.3, 2003.

[3] ALTERA Corp., "Stratix Device Handbook, Vol. 1-3", 2004.

Chapter 5

Performance analysis

In this chapter we show that DCP actually satisfies design requirements. In the first section, we discuss a mathematical modeling of the DCP system. In the second section, we discuss measured results of a DCP system and analyze the results based on the mathematical model. In the last section, we compare TCP and DCP in terms of performance.

5.1 Modeling of the DCP system

In this section, we construct a mathematical model of the DCP system with queuing theory [1]. The system parameters which characterize a DAQ system are an average data length in a buffer of senders (members), an available input rate of senders (members), and an average transfer time. By this mathematical model, we show that these parameters can be derived from a few measurable parameters.

In the first subsection, we introduce queuing theory by constructing a mathematically simplified model of a sender. In the second subsection, we construct a model of a DCP system. In the last two subsections, we extend the model to a generalized DCP system.

5.1.1 Model of a sender

At the beginning of discussions about mathematical modeling of a DCP system, we introduce queuing theory and its terminology with an example of a general single sender system. We note that the system is not a DCP system.

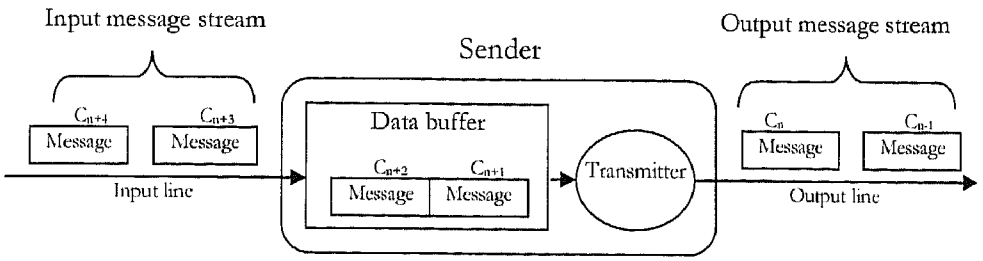

Figure 5.1: A sender system.

Figure 5.1 shows the example system, which consists of an input line, a data buffer, and an output line. In the figure, C_n represents the n-th message to enter this system, which is used in later discussions. The purpose of the sender is to transfer input messages to a receiver. The input message stream is generated by an external system such as a detector subsystem. The stream consists of messages, which are displayed in the figure. We assume that a length of all messages is one bit. An input message goes into the data buffer. At that time, if the transmitter processes no message, the message is taken from the buffer and the sender transmits the message into the output line. If the message is not the first data in the buffer or the transmitter is processing another message, the message is forced to wait while the sender is busy to transmit other messages. In this situation, a waiting line is made in the data buffer by the waiting to transmit other messages. When the waiting line is made, an order of outgoing messages from the data buffer is the same order of their arrivals.

We discuss this system in terms of queuing theory. Figure 5.2 shows this system that consists of an input line, a queue, a server, and an output line. The queue represents the data buffer of this system, which is a waiting line. The server represents the transmitter and processes messages in one by one.

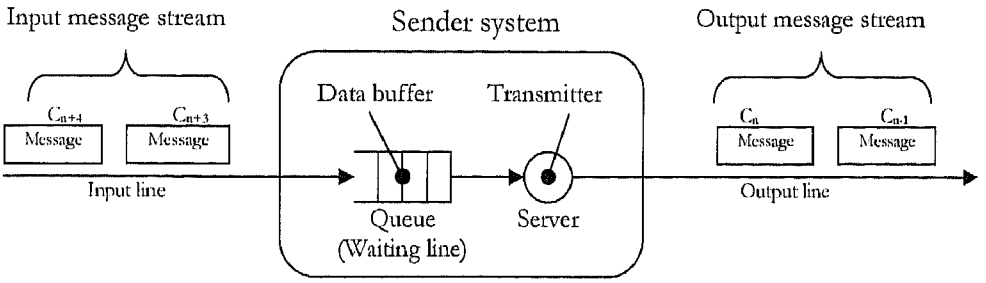

Figure 5.2: Queuing system

We focus attention on the flow of messages as they arrive, pass though, and eventually leave this system. We begin with the definitions. Recalling the n-th message is denoted by C_n, we define its arrival time to this system as

$$\tau_n \equiv \text{arrival time for } C_n . \tag{5.1}$$

And we further define a inter-arrival time between C_{n-1} and C_n as

$$a_n \equiv \text{inter - arrival time between } C_{n-1} \text{ and } C_n . \tag{5.2}$$

We assume that all inter-arrival times are independent, identically distributed random variables. Its probability distribution function (PDF) is denoted by $A(t)$, and we have

$$\mathbf{P}[a_n \le t] = A(t). \tag{5.3}$$

Here, $\mathbf{P}(x)$ is probability of an event x occurred. In this thesis, we employ Poisson arrival process. Therefore, this PDF is independent of n and is given by

$$A(t) = 1 - e^{-\lambda t}, \ t \ge 0 . \tag{5.4}$$

Here, λ [bit/s] is the arrival rate. Similarly, we define a service time for C_n as

$$s_n \equiv \text{service time for } C_n . \tag{5.5}$$

We assume that all service times are independent and identically distributed random variables. Its PDF is denoted by $S(t)$ and we have

$$\mathbf{P}[s_n \le t] = S(t). \tag{5.6}$$

This PDF is independent of n. Then its probability density function is

$$f_s(t) = \frac{dS(t)}{dt} . \tag{5.7}$$

We also assume existence of its limiting random variable that is denoted by s. We define a waiting time that is a time spent in the queue as

$$w_n \equiv \text{waiting time in queue for } C_n . \tag{5.8}$$

We assume existence of its limiting random variable that is denoted by w. We define a transfer time that is the total time spent in the system by C_n, which is the sum of its waiting time and service time. The transfer time is represented by

$$T_n \equiv w_n + s_n . \tag{5.9}$$

We assume existence of its limiting random variable that is denoted by T. We define the queue length that is the number of messages in the queue as

$$L_q(t) \equiv \text{number of messages in the queue at time } t . \tag{5.10}$$

We define the number of messages in the server as

$$L_s(t) \equiv \text{number of messages in the server at time } t. \tag{5.11}$$

In our system, this value is zero or one because our system is a single server system. The number of messages in this system is the sum of the number of messages in the queue and that in the server, which we denote by

$$Q(t) \equiv L_q + L_s. \tag{5.12}$$

Finally we define a server utilization factor as

$$\rho \equiv \lambda \bar{s}. \tag{5.13}$$

Here, $\bar{s}$ is an average of the service times. This factor may be interpreted as the probability that the server is busy at randomly selected times [1].

With the above notation we introduce a time-diagram for the queue, which permits a graphical view of the dynamics of our system. The diagram is shown in Figure 5.3.

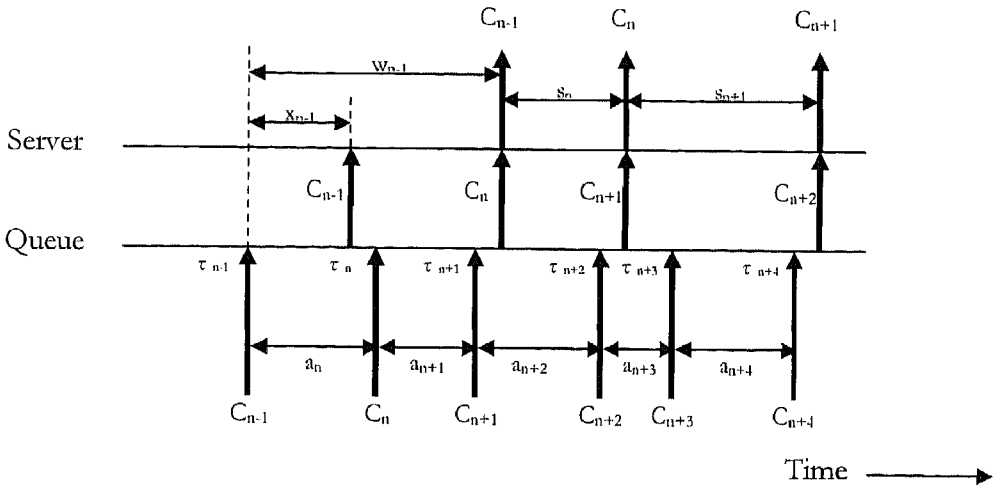

Figure 5.3: Time-diagram notation

We introduce Little's law for later discussions. It relates the average number of messages in a queuing system, the average arrival rate to the queuing system, and the average waiting time to get through the queuing system. The law is represented by

$$\bar{Q} = \lambda \bar{T},$$

$$\overline{L_q} = \lambda \bar{w}. \tag{5.14}$$

Here, $\overline{Q}$ is the average number of messages in the system and $\overline{T}$ is the average of transfer times.

Next we analyze this system under the following condition:

- Single server,
- Poisson arrival process,
- The queue length is infinite,
- The PDF of service times is a general distribution.

In queuing theory, this queuing system is denoted by M/G/1. We can exactly solve this system with queuing theory [1] but the theory use technical methods to solve it. Then, it is difficult to intuitive understand the behavior. Therefore, in this subsection, we intuitively analyze this system and derive an average transfer time and so on.

We will derive the average waiting-time. We consider the system when a message arrives to the queue. In this situation, we can consider two status of the server, which are the idle and busy states. If the server is the idle state, the data is immediately processed then the waiting time is zero. If the server is in busy state, the message is forced to wait and the waiting time is decomposed into two parts which are a residual service time and a waiting time without oneself. We denote the residual service time and waiting time are t_r and t_w, respectively. On the other hand, the probability of busy state of the server is represented by the server utilization factor ρ and the probability of the idle state is $1-\rho$. Therefore, the average waiting-time is represented by

$$\begin{aligned}\overline{w} &= \rho \times \mathbf{E}(t_r + t_w) + (1-\rho) \times 0 \\ &= \rho \times (\overline{t_r} + \overline{t_w})\end{aligned} \tag{5.15}$$

Here, $\overline{w}$ is the average of w, $\mathbf{E}(t_x + t_w)$ is an average of t_r+ t_w. Since $\overline{t_w}$ is the average waiting-time, it is equal to $\overline{w}$.

$$\overline{t_w} = \overline{w}. \tag{5.16}$$

Next we analyze $\mathbf{E}(t_r)$ that is represented by

$$\overline{t_r} = \int E(t_r|y) f_y(t) dt. \tag{5.17}$$

Here, y is a selected service time that is met a message arrival to the queue, $\mathbf{E}(t_r|y)$ is a conditional expectation value of t_r when the selected service time is y, $f_y(t)$ is a probability density function of a selected service time. We discuss the selected service

time in more detail. The arrival is Poisson process and messages randomly arrive to the queue. Therefore, we recognize that the probability that an interval length of a service time, s, is chosen should be proportional to the length as well as to the relative occurrence of such intervals. Thus, for the selected service time, we may write

$$f_y(t)dt = Ktf_s(t)dt\,. \tag{5.18}$$

Here, K is a constant for the probability normalization. We integrate the above equation to obtain K.

$$\begin{aligned} \int f_y(t)dt &= K\int tf_s(t)dt \\ 1 &= K\bar{s} \\ \Rightarrow K &= \frac{1}{\bar{s}}\,. \end{aligned} \tag{5.19}$$

On the other hand, the residual service time is randomly selected within the selected service time and arrival times of messages might be uniformly distributed over the selected time. Therefore, when the length of selected interval is X, we obtain the average of t_r is

$$\mathbf{E}(t_r|X) = \frac{X}{2}\,. \tag{5.20}$$

Combining equations (5.18), (5.19) and (5.20), we obtain the following equation:

$$\begin{aligned} \bar{t_r} &= \int \mathbf{E}(t_r|Y) f_y(t)dt \\ &= \int \frac{t}{2}\frac{tf_s(t)}{\bar{s}}dt \\ &= \frac{1}{2\bar{s}}\int t^2 f_s(t)dt \\ &= \frac{\overline{s^2}}{2\bar{s}} \end{aligned} \tag{5.21}$$

Here, $\overline{s^2}$ is the second moment of s.

Substituting equations (5.16) and (5.21) into equation (5.15), we obtain the final result as

$$\overline{w} = \rho \times \left[\frac{\overline{s^2}}{2\overline{s}} + \overline{w} \right],$$

$$\Rightarrow$$

$$\begin{aligned} \overline{w} &= \frac{\rho \overline{s^2}}{2(1-\rho)\overline{s}} \\ &= \frac{\lambda \overline{s^2}}{2(1-\rho)} \end{aligned} \quad . \tag{5.22}$$

By the definition of the transfer time, the its average is

$$\begin{aligned} \overline{T} &= \overline{s} + \overline{x} \\ &= \overline{s} + \frac{\lambda \overline{s^2}}{2(1-\rho)} \end{aligned} \quad . \tag{5.23}$$

Since the average waiting time is derived, we can also derive the average number of messages in the system with Little's law as

$$\begin{aligned} \overline{Q} &= \lambda \overline{T} \\ &= \lambda \overline{s} + \frac{\lambda^2 \overline{s^2}}{2(1-\rho)} \end{aligned} \quad . \tag{5.24}$$

5.1.2 Simplified model of a DCP system.

In this subsection, we discuss a modeling of a DCP system. We use the definitions which are defined in the previous subsection. Recall that the DCP system consists of a collector (receiver) and members (senders). In the system, transmissions of members are controlled by a token passing mechanism and an only one member that has a token packet is permitted to transmit the data to the collector. Therefore, the sender to transmit data is switched by a token. We model the mechanism with a walking server whose motion represents the token packet.

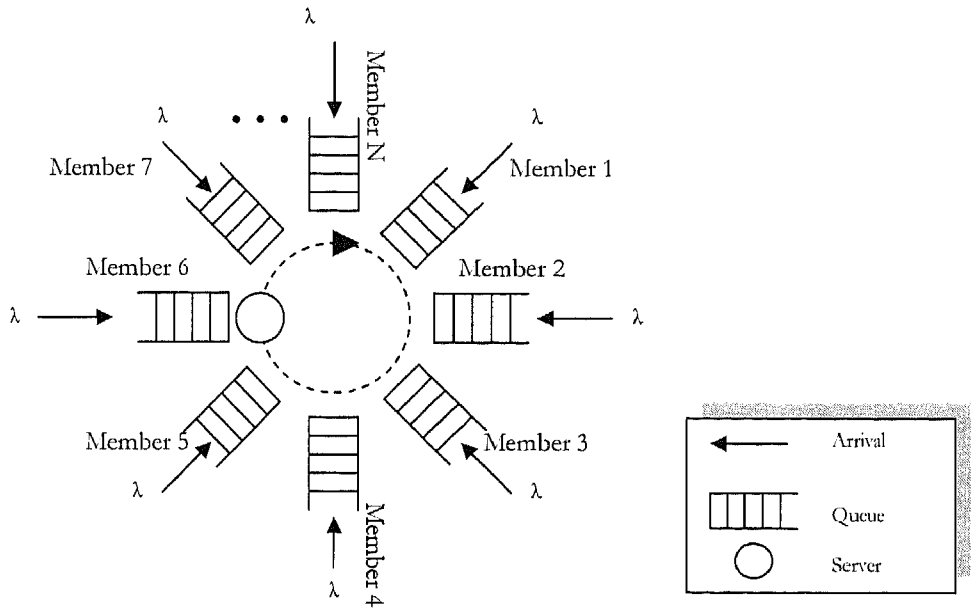

Figure 5.4: DCP system model

Figure 5.4 shows a model of a DCP system, which consists of N members and the walking server. The i ($1 \leq i \leq N$) represents the member number. We assume that the queue capacity of each member is infinite for simplicity. The arrival process at each member is independent and Poisson process with arrival rate λ [bit/s]. In the model, the server accesses multiple queues in cyclic order. Only those messages which are found at the server arrival to a member are served. The messages which arrive to the system during the servicing period are reserved to be serviced in the next time. Therefore, a length of served data is decided at the server arrival to the member and we call the length a message length, L [bit]. Strictly speaking, the message length has an upper limit in the DCP system but we assume that the message length has no limit for simplicity. We assume that all service times of members have the same PDF whose random variable is denoted by s [s]. We introduce a new variable that is a busy period, b [s]. The busy period is defined as a time interval during which the server continues to serve the data whose length is the message length. The server spends time to walk from a member to the next member. We call the spent time a switchover time and assume that all switchover times between members are random variables, independent of any of the other parameters, and have the same PDF. The random variable is denoted by u [s]. Figure 5.5 shows a relation of service times, a busy period, and a switchover time.

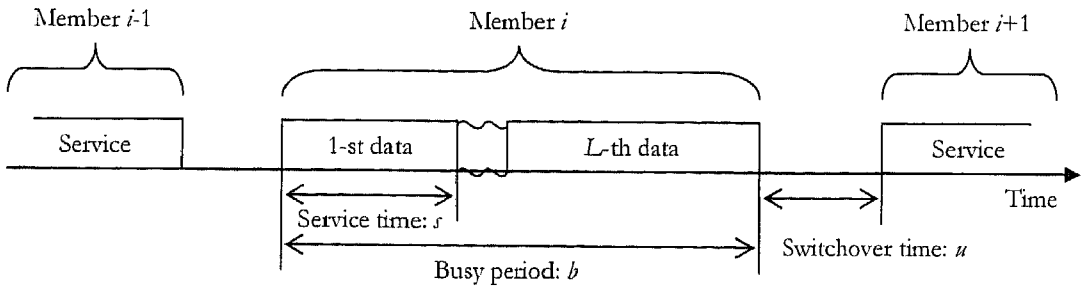

Figure 5.5: Relation of service, busy and switchover times.

The system model as above discussed has been studied in queuing theory as a polling model [2] [3]. In our model, all members have the same arrival and service conditions, which is called a symmetric system in queuing theory. The polling model is exactly solved in papers [4] [5] (Appendix B). Here, we then intuitionally solve and the results are identical to the exact solutions. The detail derivations of the exact solutions are discussed in Appendix B.

Since we would discuss the model under stability operation, we require the system to satisfy the following condition at each member,

Arrival rate = Service rate,

$$\lambda = \frac{\bar{L}}{\bar{c}}. \tag{5.25}$$

Here, $\bar{L}$ represents an average message length and $\bar{c}$ represents an average cycle time. The cycle time is the time interval between the time when a server arrived to a member and the time when the server arrived again to the member. From definitions, the average cycle time $\bar{c}$ is represented by

$$\begin{aligned} \bar{c} &= \sum_{i=1}^{N} \left(\bar{u} + \bar{b}\right) \\ &= N\left(\bar{u} + \bar{b}\right) \end{aligned}. \tag{5.26}$$

Here, $\bar{u}$ represents an average switchover time and $\bar{b}$ represent an average busy period of a member, which is represented by

$$\bar{b} = \bar{L}\bar{s}. \tag{5.27}$$

Rewriting equation (5.26) with the above equation, thus

$$\bar{c} = N\bar{u} + N\bar{L}\bar{s}. \tag{5.28}$$

Therefore, substituting $\bar{L}$ of equation (5.25) into equation (5.28) gives the average cycle time:

$$\bar{c} = \frac{N\bar{u}}{1 - N\lambda\bar{s}}. \tag{5.29}$$

We defined the server utilization factor in the previous subsection. We define the server utilization factor of a member in a similar fashion by

$$\rho_i \equiv \lambda\bar{s}, \quad i=1,\cdots,N. \tag{5.30}$$

Here, i is a member number. In the model, since a factor, $N\lambda\bar{s}$, is often appeared in equations, we define the factor as a total server utilization factor, ρ, which is formally defined as

$$\begin{aligned} \rho &\equiv \sum_{i=1}^{N} \rho_i \\ &= N\lambda\bar{s} \end{aligned}. \tag{5.31}$$

Rewriting equation (5.29) with the above definition, we obtain the final expression of the average cycle time, thus

$$\bar{c} = \frac{N\bar{u}}{1-\rho}. \tag{5.32}$$

We can also obtain the average message length in a similar fashion. Substituting $\bar{c}$ of equation (5.25) into equation (5.28) gives the average message length.

$$\bar{L} = \frac{\lambda N\bar{u}}{1-\rho}. \tag{5.33}$$

We note that the average message length is not an average queue length at any time and is an average queue length at the server arrival to the member.

Next we discuss an average waiting time. When a message arrivals to a member, the server is walking between members or servicing data. Therefore, the average waiting time is represented by

$$\bar{w} = \mathbf{P}(S_m)\overline{x_m} + \mathbf{P}(S_s)\overline{x_s}. \tag{5.34}$$

Here, $\mathbf{P}(S_m)$ and $\mathbf{P}(S_s)$ are probabilities of the server in moving and servicing, respectively; $\overline{x_m}$ and $\overline{x_s}$ are average waiting times at the server moving and servicing

when the message arrives to the member, respectively. $\mathbf{P}(S_m)$ can be represented with the average cycle and switchover times, thus

$$P(S_m) = \frac{N\bar{u}}{\bar{c}}. \tag{5.35}$$

$\mathbf{P}(S_s)$ is also represented, thus

$$P(S_s) = \frac{\bar{c} - N\bar{u}}{\bar{c}}. \tag{5.36}$$

We discuss, $\overline{x_m}$, the average waiting time at the server moving when the message arrives to the member. We can decompose the average into two parts: one is an average time of a residual switchover time, $\overline{x_m^r}$, and the other is an average cycle time from the viewpoint of the message, $\overline{x_m^c}$.

$$\overline{x_m} = \overline{x_m^r} + \overline{x_m^c}. \tag{5.37}$$

In the previous subsection, we derive the average of residual service times. We can calculate the average of residual switchover times in a similar fashion, thus

$$\overline{x_m^r} = \frac{\overline{u^2}}{2\bar{u}}. \tag{5.38}$$

We discuss the cycle time from the viewpoint of the message. The cycle time is shorter than c by the selected switchover time u. Moreover, we should add a busy period because a message is forced to wait until the server is finish to transmit other messages when the server arrival to the member. On the other hand, since the arrival process is Poisson process, the arrival times are uniformly distributed over the cycle time. Therefore, we should multiply 1/2 to average the cycle time. Finally, we can represent $\overline{x_m^c}$ by

$$\overline{x_m^c} = \frac{1}{2}\left[\left(\bar{c} - \bar{u}\right) + \bar{b}\right]. \tag{5.39}$$

We discuss, $\overline{x_s}$, another average in equation (5.34); the average waiting time at the server servicing when the message arrives to the member. This situation is equivalent to the single queue system that is discussed in the previous subsection. Therefore, the average is represented by

$$\overline{x_s} = \frac{\overline{s^2}}{2\bar{s}} + \bar{w}. \tag{5.40}$$

Combining equations (5.34) through (5.40), we obtain the following equation:

$$\bar{w} = \frac{N\bar{u}}{\bar{c}}\left[\frac{\overline{u^2}}{2\bar{u}} + \frac{1}{2}\left(\bar{c} - \bar{u} + \bar{b}\right)\right] + \frac{\bar{c} - N\bar{u}}{\bar{c}}\left(\frac{\overline{s^2}}{2\bar{s}} + \bar{w}\right). \tag{5.41}$$

We rewrite the above equation with equations (5.27), (5.32), and (5.33):

$$\begin{aligned} \bar{w} &= (1-\rho)\left[\frac{\overline{u^2} - \bar{u}^2}{2\bar{u}} + \frac{1}{2}\frac{N\bar{u}}{1-\rho}\left(1 + \lambda\bar{s}\right)\right] + \rho\left(\frac{\overline{s^2}}{2\bar{s}} + \bar{w}\right) \\ \Rightarrow \bar{w} &= \frac{\overline{u^2} - \bar{u}^2}{2\bar{u}} + \frac{1}{2(1-\rho)}\left[N\bar{u}\left(1 + \lambda\bar{s}\right) + \rho\frac{\overline{s^2}}{\bar{s}}\right] \end{aligned} \tag{5.42}$$

Finally, we obtain the average transfer time as the following equation:

$$\bar{T} = \bar{s} + \frac{\sigma_u^{\,2}}{2\bar{u}} + \frac{1}{2(1-\rho)}\left[N\bar{u}\left(1 + \lambda\bar{s}\right) + \rho\frac{\overline{s^2}}{\bar{s}}\right]. \tag{5.43}$$

Here, $\sigma_u^{\,2}$ represents the variance of a total switchover time. We can also obtain the queue length with Little's law.

$$\bar{Q} = \lambda\left\{\bar{s} + \frac{\sigma_u^{\,2}}{2\bar{u}} + \frac{1}{2(1-\rho)}\left[N\bar{u}\left(1 + \lambda\bar{s}\right) + \rho\frac{\overline{s^2}}{\bar{s}}\right]\right\}. \tag{5.44}$$

We note that the necessary condition for system stability is given by [6]

$$\rho < 1. \tag{5.45}$$

In the next section, we will use the second moment of message lengths then to experimentally verify the model. The second moment is given by

$$\overline{L^2} = \frac{1}{(1-\rho)(1+\lambda\bar{s})}\left[N\lambda^2\overline{u^2} + \frac{N^2\lambda^3\bar{u}}{(1-\rho)}\overline{s^2} + N\lambda^2\bar{u}^2\left(\frac{N(1+\rho)}{(1-\rho)} - 1\right)\right] + \bar{L}. \tag{5.46}$$

The derivation of the above equation is discussed in Appendix B.

Since we assume that we can design the network of a DCP system, we can know values of parameters without switchover times. The switchover times depend on network devices, for example, HUB. Therefore, we should measure the switchover times to calculate the average and moment values. That is, if we measure only the switchover times then the average and moment values can be calculated.

We summarize the equations as follows:

The average message length:

$$\overline{L} = \frac{\lambda N \overline{u}}{1-\rho}, \tag{5.47}$$

The second moment of message length:

$$\overline{L^2} = \frac{1}{(1-\rho)(1+\lambda\overline{s})}\left[N\lambda^2 \overline{u^2} + \frac{N^2\lambda^3 \overline{u}\,\overline{s^2}}{(1-\rho)} + N\lambda^2 \overline{u}^2 \left(\frac{N(1+\rho)}{(1-\rho)} - 1 \right) \right] + \overline{L}, \tag{5.48}$$

The average transfer time:

$$\overline{T} = \overline{s} + \frac{\sigma_u^{\,2}}{2\overline{u}} + \frac{1}{2(1-\rho)}\left[N\overline{u}(1+\lambda\overline{s}) + \rho\frac{\overline{s^2}}{\overline{s}} \right]. \tag{5.49}$$

5.1.3 Single HUB system

In this subsection, we analyze a single HUB system in order to discuss the model in more detail. The system under consideration is shown in Figure 5.6. There are N members, a HUB, and a collector.

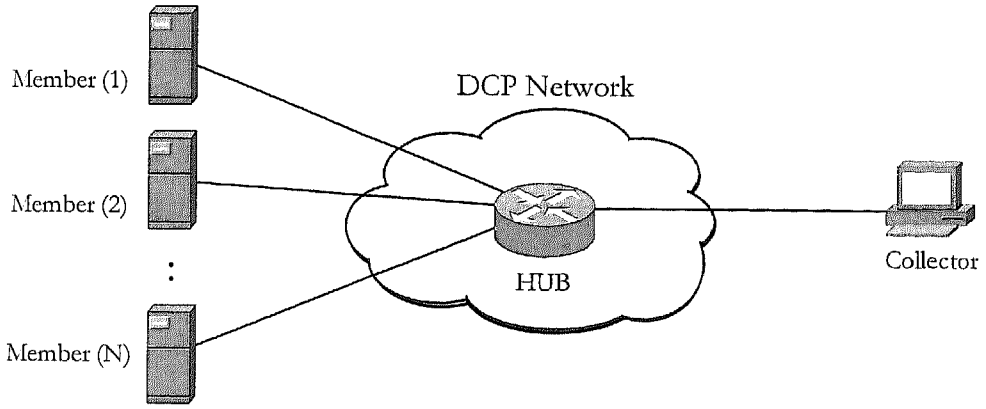

Figure 5.6: Basic DCP system.

We should measure the switchover times to calculate the average and moment values, which is pointed out in the previous subsection. In order to simplify to measure it, we measure a period of a token spent time to go around a logical token-ring without transmitting data packets. We call the period a free-token round-trip-time (FTRTT) that is denoted by F [s]. An average FTRTT plays an important role in a performance analysis of a DCP system because we can calculate the average and moment values if we measure FTRTT's.

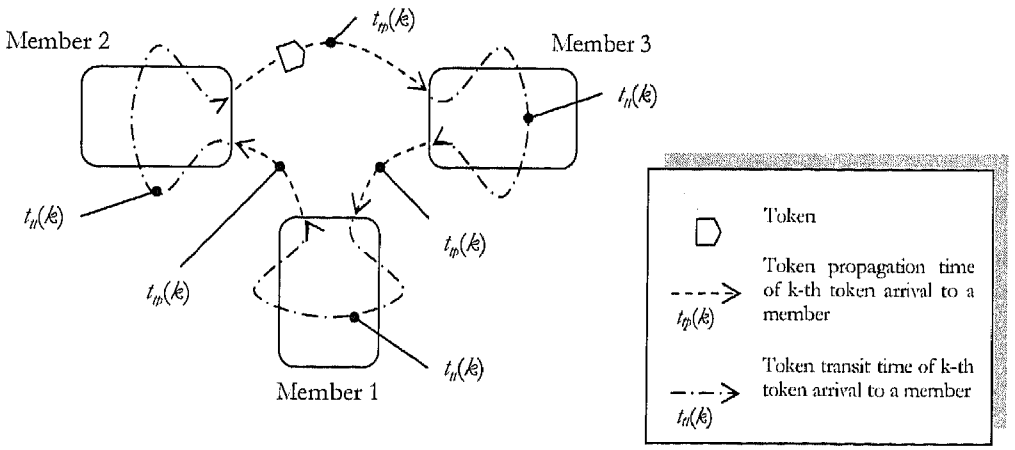

Figure 5.7: Free token round trip time

A case of three members system is shown in Figure 5.7. The FTRTT is decomposed into two parts, one is a token-propagation time between members, and the other is a token-transit time in members. The token-propagation time is the time interval from the transmitted time of a token packet at member i to the time when the packet is received at member i+1. The token-transit time is the processing period of the token packet. We denote $t_{tp}(k)$ [s] and $t_{tt}(k)$ [s] by the token-propagation time and the token-transit time of k-th token arrival to a member, respectively. Recall that our model is a symmetric system and we analyze it. Therefore, we assume all the token-propagation times have the same PDF and the token-transit times also have the same PDF. An average FTRTT is represented with an average token-propagation time and an average token-transit time, thus

$$\overline{F} = N\left(\overline{t_{tt}} + \overline{t_{tp}}\right). \tag{5.50}$$

In the case of the single HUB system, the average FTRTT is exactly equal to the average total switchover time, thus

$$\overline{F} = N\overline{u}. \tag{5.51}$$

The average FTRTT is not always equal to the average total switchover time but the average and moment values can be calculated if we measure only FTRTT's, which is discussed in the next subsection.

Next we discuss the service time that is introduced in the previous subsection. Here, we discuss the times in more detail. In the DCP system, the data are transmitted by packets. We have defined the busy period as a time interval during the server continues to serve messages. In DCP systems, the period is a time during the member transmits data packets to a collector, which can be decomposed into a message preparing and two transmitting periods which are for the DCP payload data and packet headers. Since our DCP system is designed that the preparing period is zero, we will omit the period in the following discussion. Moreover, the transmitting period can be decomposed into service times.

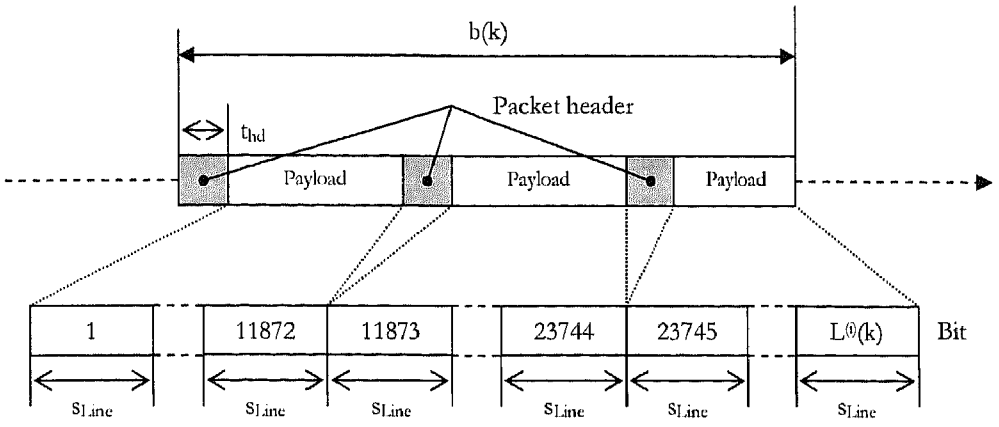

Figure 5.8: The busy period at the k-th token arrival to a member.

Figure 5.8 shows the busy period at the k-th token arrival to a member; $b(k)$ [s], $L(k)$ [bit], s_{Line} [s], and t_{hd} [s] represent the busy period, the message length, the transmitting period of one bit, and the period to process a packet header, respectively. Since we are discussing a symmetric system, the s_{Line} for all members is the same value and is an inverse value of the line interface speed of the member, for example, when we employ Fast Ethernet [7] that value is 10 ns. Therefore, the busy period of a single HUB system is

$$\begin{aligned}\bar{b} &= t_{hd} \times \left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right] + \bar{L}s_{Line} \\ &= s_{Line} \times \left\{L_{hd}\left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right] + \bar{L}\right\}.\end{aligned} \tag{5.52}$$

Here, P_{max} [bit] is a maximum DCP payload length and L_{hd} [bit] is a header length of a packet. When we employ Ethernet for a network, P_{max} is 11872 bits and L_{hd} is 336 bits. We discuss the service time again. The service time is defined by a period to service a unit data, its average is represented with the busy period, thus

$$\begin{aligned}\bar{s} &= \frac{\bar{b}}{\bar{L}} \\ &= s_{Line}\left(\frac{L_{hd}\left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right]}{\bar{L}}+1\right).\end{aligned} \tag{5.53}$$

The second moment of the service time is represented by

$$\overline{s^2} = \frac{S_{Line}{}^2\left[\bar{L}-\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)-1\right]+\left(S_{Line}L_{hd}\right)^2\left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right]}{\bar{L}}. \tag{5.54}$$

We summarize the equations as follows:

The average service time:

$$\bar{s} = s_{Line}\left(\frac{L_{hd}\left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right]}{\bar{L}}+1\right), \tag{5.55}$$

The second moment of service time:

$$\overline{s^2} = \frac{S_{Line}{}^2\left[\bar{L}-\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)-1\right]+\left(S_{Line}L_{hd}\right)^2\left[\mathrm{mod}\left(\bar{L}-1, P_{\max}\right)+1\right]}{\bar{L}}, \tag{5.56}$$

The total switchover time of a single HUB system:

$$\overline{F} = N\bar{u}, \tag{5.57}$$

5.1.4 Multiple HUB system

In this subsection, we discuss a multiple HUB system. Packets pass only one HUB in a single HUB system but packets will pass multiple HUB's in a multiple HUB system. In this case, since a token packet and data packets are forwarded to the same port of a HUB, the token packet is forced to wait until the service completion of data packets which are transmitted before the time when the token packet is transmitted.

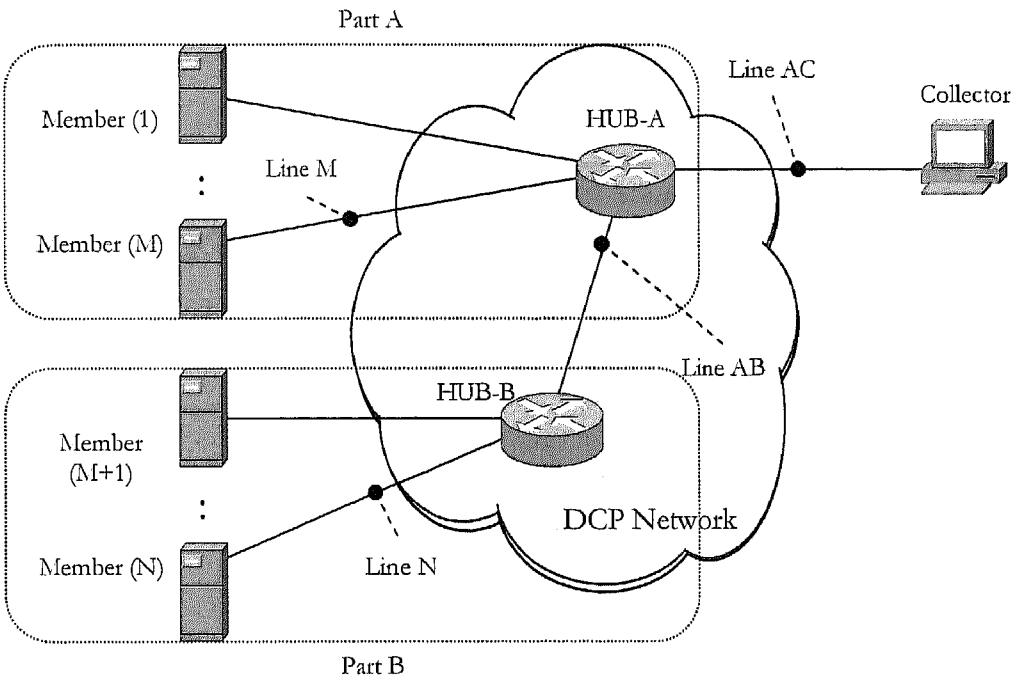

Figure 5.9: Multiple HUB system.

We discuss the multiple HUB system by considering an example system that is shown in Figure 5.9. There are N members, two HUB's, and a collector. Firstly, we would discuss a packet switching behavior in the part-A, and next in the part-B. In the part-A, the situation of this part is the same as a single HUB system. When member M in part-A transmits packets, packets are switched as illustrated in Figure 5.10. In the figure, we ignore packet switching times. The member transmits a data packet whose length is longer than a token packet and, after that time, the token packet is transmitted. Since we assume that a packet switching method of all HUB's in a DCP system is a store and forward method, the data packet is started to transmit to line-AC. Next the member transmits the token packet to member M+1. The token packet is forwarded in a

similar fashion. In this situation, we note that the time when the token packet is started to transmit to line-AB is earlier than the time when the data packet is finished to transmit to line-AC. In this case, a forwarding timing of the token packet is not influenced by data packets.

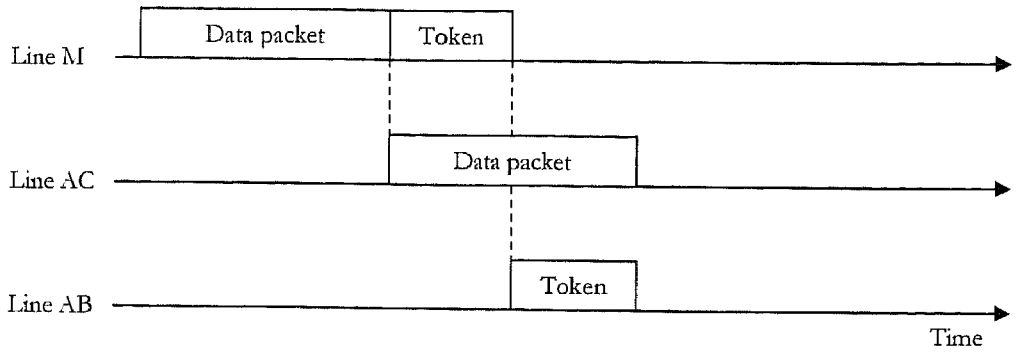

Figure 5.10: Packet switching behavior in part-A.

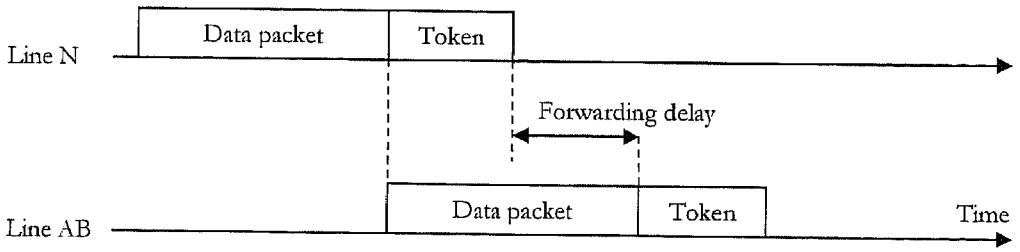

Figure 5.11: Packet switching behavior in the part B.

We consider behavior of packet switching in part-B. When member N in part-B transmits a data packet to the collector and the token packet to member 1, the packets are switched as illustrated in Figure 5.11. This behavior is different from the case of part-A. The token packet is delayed by the data packets. Clearly, this situation is different from the condition when FTRTT's are measured. If a length of data packet is equal or smaller than a length of a token packet, the delay does not occur. The points over a token-ring where the delay occurs are called the token-delay points. We note that the number of token delay points depends on a sequence of a token ring.

As discussed above, we find that a switchover time depends on a length of data packets. When the total number of data packets in a busy period is one, the switchover time is decided by the length of the data packet. When DCP sends data packets, all length of data packets without the last data packet are the same length which is the maximum payload length. Therefore, if the total number is two or more, the switchover time is decided by the maximum payload length. The total switchover time is represented by

$$\begin{aligned} \overline{u} &= \overline{F} && ,(0 \le \overline{L} \le P_{\min}); \\ &= \overline{F} + s_{Line} D\left(\overline{L} - P_{\min}\right) && ,(P_{\min} < \overline{L} < P_{\max}); \\ &= \overline{F} + s_{Line} D\left(P_{\max} - P_{\min}\right) && ,(P_{\max} \le \overline{L}). \end{aligned} \tag{5.58}$$

Here, D, P_{min}, and P_{max} are the number of token-delay points, the minimum and maximum DCP payload lengths, respectively. When a system has only one HUB, the number of token delay points is always zero.

5.2 Experimental verification

We construct the model in the previous section. We verify the model by experiments. In this section, we discuss measured results of DCP systems and analyze these results.

5.2.1 Setup

In order to measure the parameters of DCP systems, we have implemented a DCP member on an FPGA and a DCP collector on Linux OS. We measure the following parameters:

- FTRTT,
- Message length, and
- Transfer rate variation.

The hardware implementation is discussed in chapter 4. The software implementation of the collector is running in a user space and it is implemented by using standard functions of the OS, *socket* functions and so on.

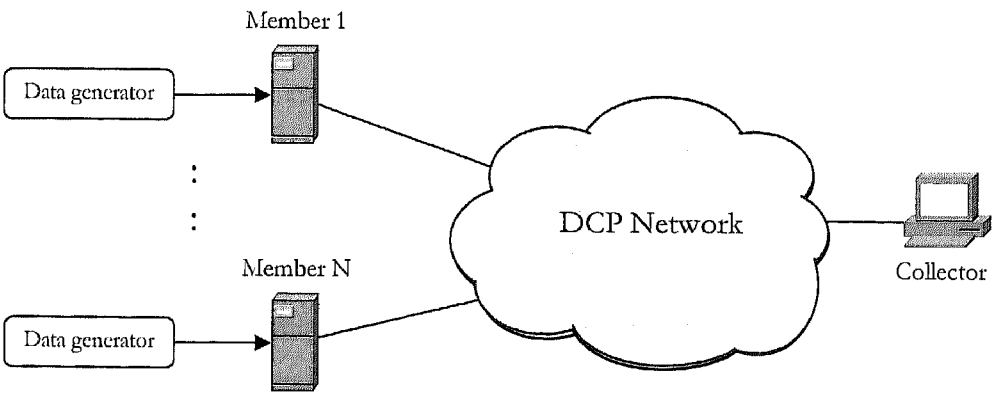

Figure 5.12: Setup for the measurements.

Figure 5.12 shows a setup for the measurements. There are N senders, a collector, and a DCP network. We measured parameters for various types of DCP networks. The data generators generate the Poisson processes and all arrival rates are λ.

The FTRTT and message length are measured at member 1 by an FPGA. The member sends a token that has the time stamp when the token is transmitted, and, then,

the packet is passed among members. We use the system clock whose frequency is 33 MHz for the time stamp. When member 1 receives the token packet, the member records the received time, the transmitted time, and a serviced message length at the token arrival.

In order to qualitative observe transfer variations of members, we make up a program. The variations are measured by the collector with software that records cumulative transferred data lengths of members by software timer interrupt that is generated every about 10 ms. We make up the program with *select* functions of the Linux OS for the generation. The interrupt does not exactly occur because the program is dependence on the OS kernel. Therefore, we design that the program records the cumulative transferred data lengths with its recoding time. However, we can not completely remove the uncertainly. Therefore, transfer rate variations which are calculated by the recorded results are sometimes zero although actually those are not equal zero but it is not a problem to qualitatively compare.

5.2.2 Single HUB system

A test bed for the single HUB system is shown in Figure 5.13, which consists of three members, a collector, and a HUB. The HUB specification is summarized in Table 5.1. Employing this test bed, we will discuss the following topics:

- Average FTRTT,
- Transfer rate variations,
- Average message length,
- Calculated average waiting-time, and
- Robustness for packet losses.

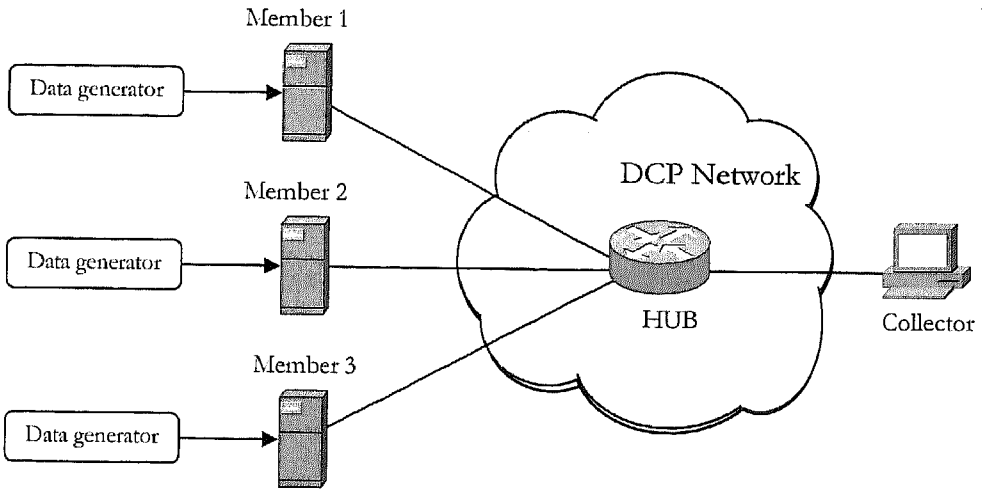

Figure 5.13: Test bed for a single HUB system.

Table 5.1:

Summary of HUB specifications.

Model	BUFFALO LSW10/100-5P
Frame switching mechanism	Store and forward
Max forwarding throughput	148810pps (100BASE-T)
Number of port	5 (RJ45)
MAC address table size	4096
Packet buffer size (byte)	128K
Physical characteristics (mm)	105(W) x 79(D) x 26(H)
Weight (g)	145

FTRTT

We discuss an average FTRTT that is one of important parameters that decide system performance. We measured the FTRTT at member 1; a FTRTT is defined as a time interval between times when token packets are transmitted at member 1. Figure 5.14 shows the result and the statistics of the measured data are summarized as Table 5.2. There are two values which are 62 and 69 μs, other values do not exist. These discrete values do not impact our calculation because the interval of these two values is too smaller than the average value. We will calculate the average message length and the average message waiting-time with this average FTRTT.

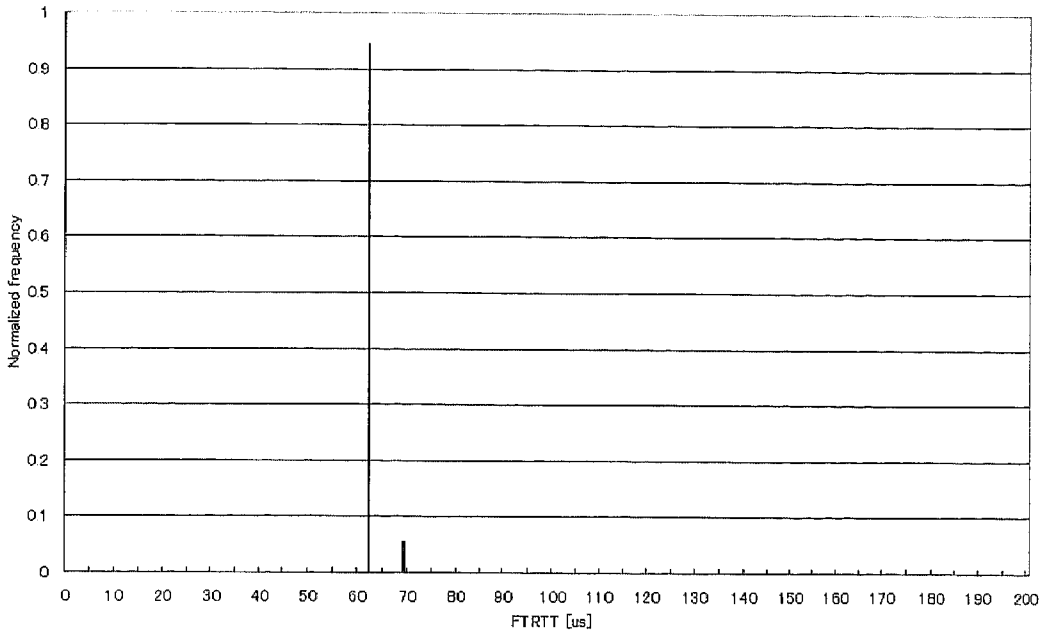

Figure 5.14: Measured FTRTT.
The normalized frequency is a frequency divided by the number of samples.

Table 5.2:
Statistics values of measured FTRTT's.

Average [μs]	Variance [μs^2]	Number of samples
62.5	3.1	9995

System parameters

We summarize system parameters of the single HUB system under measuring in Table 5.3. When we calculate the average message length and the average message waiting-time, we use the following values:

Table 5.3:
System parameters of the single HUB system.

Parameter	Value
Number of members: N	3
Number of token delay points: D	0
Transmitting period of a bit to a line: s_{Line}	10 ns
Maximum DCP payload length: $P_{\max}$	11872 bits (1484 bytes)
Minimum DCP payload length: $P_{\min}$	240 bits (30 bytes)
Average FTRTT: $\overline{F}$	63.0 μs
Header length: L_{hd}	336 bits

Transfer rate variations

We design that DCP fairly transfers data among members with a token passing mechanism. In order to confirm this fair data transfer, we measured variations of transfer rate of each member. Due to this fair data transfer, we expected that all transfer rates from members to the collector are to be nearly equal bandwidth occupancy for the DCP payload data.

Figure 5.15 shows results at λ=30 Mbit/s. The results show that all members fairly transfer data and each transfer rate variations are the nearly equal.

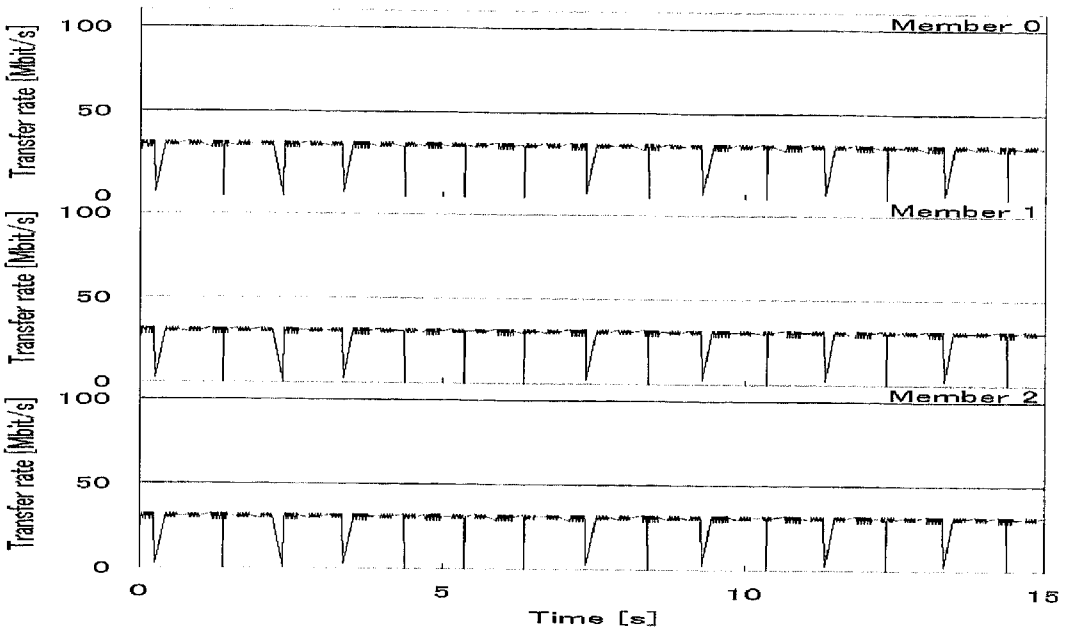

Figure 5.15: Transfer variations at λ=30 Mbit/s.

The average and second moment of message length

We measured an average message length at member 1. We calculate the average message length and the second moment of message lengths under the system parameters which are summarized in Table 5.3. The average message length is calculated with equation (5.47) and equation (5.55). The second moment of message length is calculated with the calculated results of the average message lengths, equation (5.56) and equation (5.48).

Measured message lengths are summarized in Table 5.4. Figures 5.16 and 5.17 show an average and a second moment of message lengths, respectively. In the figures, the curve and squares display a calculated values and measured values, respectively; and the dotted line displays an instability input rate, which is calculated by the stability condition equation (5.45), definition (5.31), N=3, and $\bar{s} = 10$ ns. The calculation results are in good agreement with measurement values. Since the value extremely grows when the arrival rate reaches to the instability rate, we should design that a system avoids near the instability point.

This result leads us to the conclusion that the model well explains the single HUB system and the system can be designed by the quantitative approach.

Table 5.4:

Measured message lengths of the single HUB system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [byte^2]	Number of samples
5	48.1	11.4	2342.6	65536
10	115.4	18.9	13671.7	65536
15	222.0	28.0	50086.5	65536
20	411.6	40.3	171048.4	65265
25	839.7	64.5	709314.5	64227
30	3712.0	187.7	1381427.0	22955

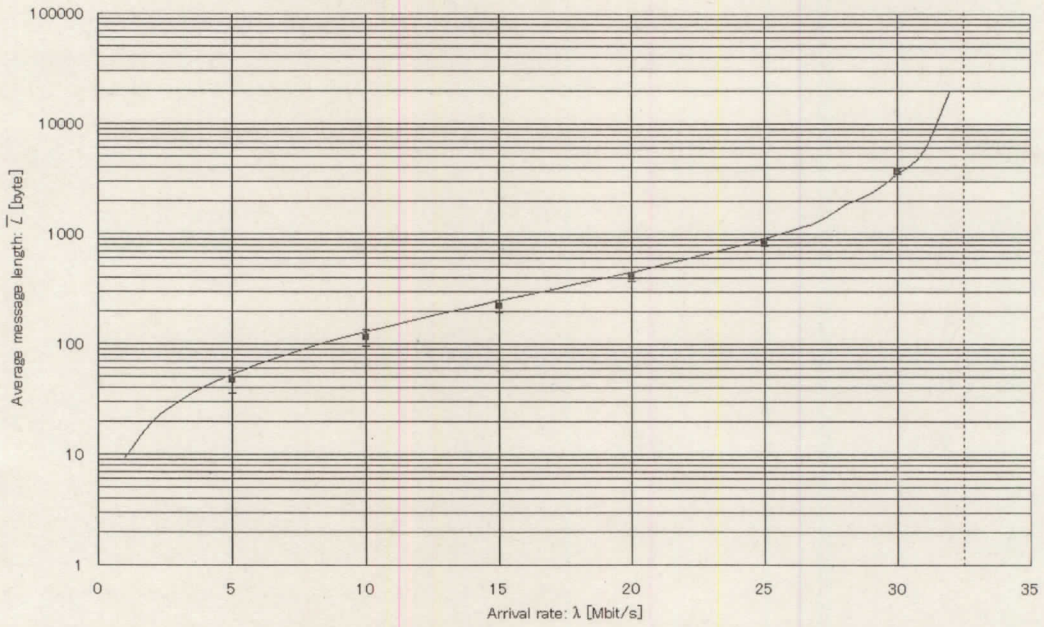

Figure 5.16: Average message length of the single HUB system.

The solid curve displays the calculated average message length, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

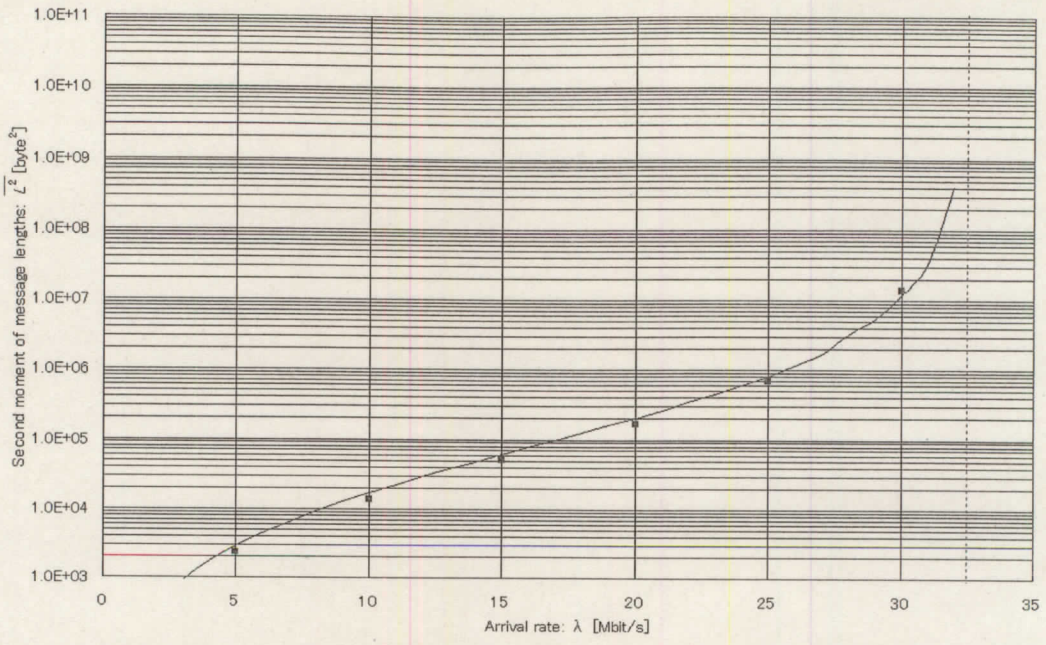

Figure 5.17: Second moment of message lengths of the single HUB system.
The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

Average transfer time

We can not measure transfer times in our system by technical problems because a length of message arrival to members is too short to stamp time information. However the average transfer time is one of important parameters when we design a DCP system. We pointed out that the model well explains the single HUB system. Therefore, we consider that an average transfer time can be calculated with the calculated result of the average message lengths, equation (5.55), and equation (5.56) under the system parameters that are summarized in Table 5.3. Figure 5.18 shows the calculated average transfer time, the curve and the dotted line display the calculated result and an instability point, respectively. This value extremely growths to reach the instability input rate. This behavior is same as the average message length.

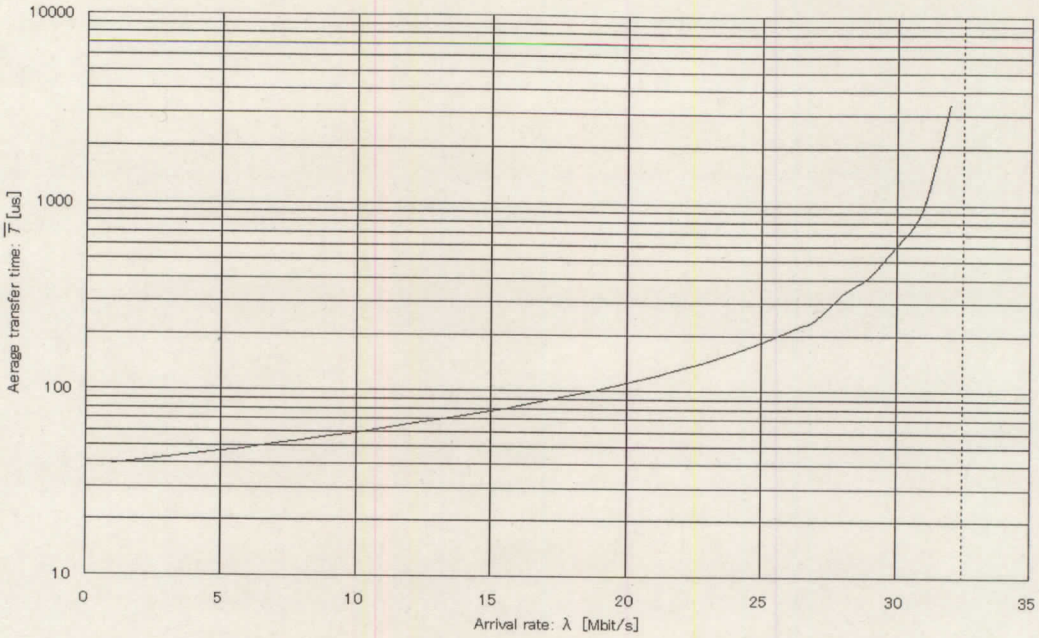

Figure 5.18: Average transfer time of the single HUB system.
The curve displays a calculated average transfer time, and the vertical dotted line displays the instability input rate.

Robustness for packet losses

The number of packet losses is very small because DCP avoids a packet loss. In the case of other measurement without this subsection, packet losses do not occur during those measuring. However, the packet loss would occur in an actual system. We then measure the robustness for packet losses.

Since DCP has a reliable data delivery, a data loss does not occur but packet losses make to decrease transfer efficiency, and, then, it finally makes to decrease the average transfer rate. In order to measure the robustness for packet losses, we modify the program to have a test function to randomly generate packet losses. We measure the average transfer rates with packet losses. All members have the same condition; input of each member is Poisson arrivals with 5 Mbit/s, 10 Mbit/s, 15 Mbit/s, 20 Mbit/s, 25Mbit/s, or 30Mbit/s; and packet loss rate of each member is 1.0%, 5.0%, 10%, 20%, 40%, or 80%.

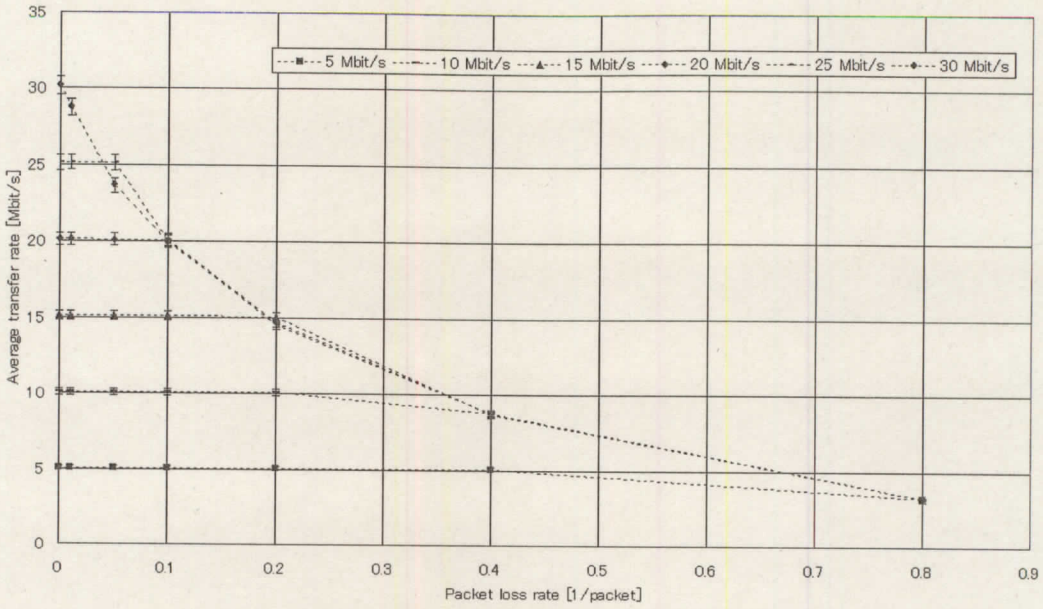

Figure 5.19: Average transfer rate with packet losses.

Figure 5.19 shows the results the average transfer rate with packet losses, an average rate is a function as a packet loss rate. There are six functions by each input data rate of a member. In the case of the input rate at 5 Mbit/s, since a used bandwidth is about 16% without packet losses, 84% of the maximum bandwidth can be used for error recovery data transfers. Therefore, the average transfer rate is kept the rate under the packet loss rate up to 40%. In the case of the input rate at 30 Mbit/s, since almost a usable maximum bandwidth is used, the average rate is decreasing from the rate 1%. At packet loss rate 5%, the average value smaller than the case of input rate 25 Mbit/s. That reason is a re-transmission mechanism for reliable data transfer and the mechanism makes many data packets due to retransmit for data recovery.

Finally, we note that if we use DCP under a situation with packet losses, we should design the maximum input rate of members by the packet loss rate.

5.2.3 Multiple HUB system

We measured parameters of multiple HUB systems and verified the model with these results. It is a difference from a single HUB system that a multiple HUB system has token-delay points.

Overview

Three test beds for the multiple HUB systems are shown in Figure 5.20, which are based on the single HUB system and are constructed to add HUB's to the single HUB system. We call these test beds the type-1, type-2, and type-3 systems, respectively. Each test bed consists of three members, a collector and HUB's. Their network topologies are different from each other: A, B, and C represent HUB names for identifying them and these specifications summarized in Table 5.5. In all systems, an order of the token passing is member 1, 2, 3, and 1 again.

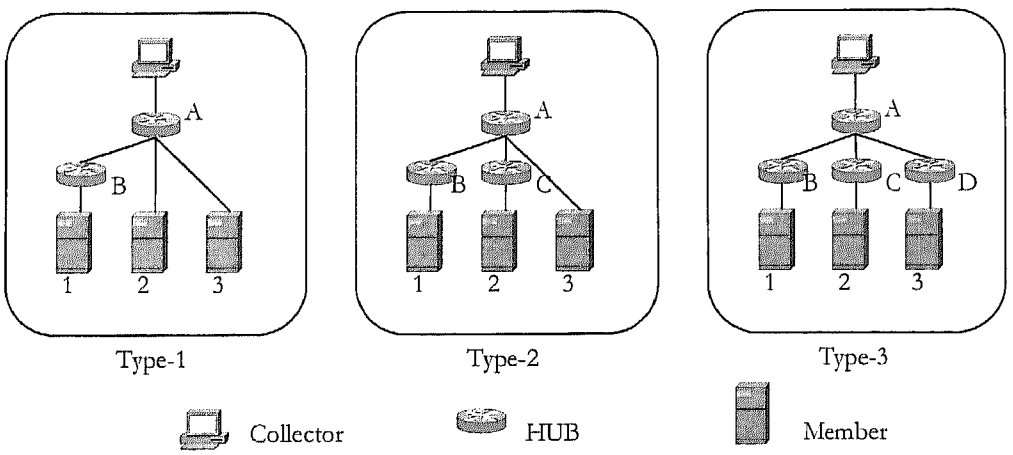

Figure 5.20: Test beds of multiple HUB systems.

Table5.5:

Summary of HUB Specifications.

	A	B, C, D
Model	BUFFALO LSW10/100-5P	BUFFALO LSW-TX-5NS
Frame switching mechanism	Store and forward	Store and forward
Max forwarding throughput	148810pps (100BASE-T)	148810pps (100BASE-T)
Number of port	5 (RJ45)	5 (RJ45)
MAC address table size	4096	1024
Packet buffer size (byte)	128K	64K
Physical characteristics (mm)	105(W) x 79(D) x 26(H)	153(W) x 83(D) x 32(H)
Weight (g)	145	430

The type-1 system consists of three members, a collector and two HUB's. This system has one token-delay point; the point is located at HUB-B and direction is from member 1 to HUB-A. The type-2 system consists of three members, a collector and three HUB's. This system has two token-delay points; one point is the same in type-1 and the other one is located at HUB-C and direction is from member 1 to HUB-A. The type-3 system consists of three members, a collector and four HUB's. This system has three token-delay points; two points are the same in type-2 and the other one is located at HUB-D and direction is from member 2 to HUB-A.

We will discuss the following topics:

- Average FTRTT,
- Average message length
- Second moment of message lengths, and
- Calculated average message waiting-time.

Recall that it is a difference between a multiple HUB system and a single HUB system that the multiple HUB system has token-delay points, which are discussed in subsection 5.1.4. At the points, a token is delayed by data packets. Therefore, we should employ equation (5.58) for the average total switchover time when we calculate system parameters.

FTRTT

We discuss the FTRTT and the value plays important roll in the multiple HUB systems. Figure 5.21 shows the measurement results of type-1 system and the statistics for the measurement are summarized in Table 5.7. There are three values which are 70,

78, and 86 μs, and other values do not exist. Figure 5.25 shows the measurement results of type-2 system and the statistics for the measurement are summarized in Table 5.9. There are two values which are 86 and 94 μs, and other values do not exist. Figure 5.29 shows the measurement results of type-3 and the statistics for the measurement are summarized in Table 5.11. There are two values which are 101 and 109 μs, and other values do not exist. In the above results, those values are discrete and the intervals between the values are 8 μs. These discrete values do not impact our calculation because the interval of these two values is too smaller than the average value.

We will calculate the average message length and the average message transfer time with these average FTRTT's. The differences of the average FTRTT of type-1, type-2, and type3 from that of the single HUB system are 14.2, 28.2, and 43.0 μs, respectively.

System parameters

We summarize system parameters of the multiple HUB system under measuring in Table 5.6. When we calculate a system parameter, we use the following values:

Table 5.6:

System parameters of multiple HUB systems.

Parameter	Value		
	Type-1	Type-2	Type-3
Number of members: N	3	3	3
Number of token delay points: D	1	2	3
Transmitting period of a bit to a line: s_{Line}	10 ns	10 ns	10 ns
Maximum DCP payload length: $P_{\max}$	11872 bits (1484 bytes)	11872 bits (1484 bytes)	11872 bits (1484 bytes)
Minimum DCP payload length: $P_{\min}$	240 bits (30 bytes)	240 bits (30 bytes)	240 bits (30 bytes)
Average FTRTT: $\overline{F}$	76.6 μs	90.7 μs	104.9 μs
Header length: L_{hd}	336 bits	336 bits	336 bits

The average and second moment of message length

We measured message lengths at member 1. We calculate the average message length and the second moment of message lengths under the system parameters which are summarized in Table 5.6. The average message length is calculated with equations (5.47), (5.55), and (5.58). The second moment of message length is calculated with the calculated results of the average message lengths, equations (5.56) and (5.48).

The measured message lengths of type-1 system are summarized in Table 5.8. The average message length of measured values and calculated results are displayed in Figure 5.22. In the similar fashion, Figure 5.23 shows the second moment of message lengths. The measurement message lengths of type-2 system are summarized in Table 5.10. The average message length of measured values and calculated results are displayed in Figure 5.26. In the similar fashion, Figure 5.27 shows the second moment of message lengths. The measurement message lengths of type-3 system are summarized in Table 5.12. The average message length of measured values and calculated results are displayed in Figure 5.30. In the similar fashion, Figure 5.31 shows the second moment of message lengths. In the figures, the solid curve, squares, and the dashed curve display the calculated result, measured values, and the calculated result of the single HUB system. And the vertical dotted line displays the instability arrival rate that is calculated by the stability condition equations (5.45), (5.31), N=3, and $\bar{s} = 10\,\text{ns}$.

We find a kink of the calculated result at L=1484 bytes in the figure. The kink comes from a large variation of the average service time. Recall equation (5.55) that is the average service time, which is

$$\bar{s} = s_{Line}\left(\frac{L_{hd}\left[\text{mod}\left(\bar{L}-1, P_{\max}\right)+1\right]}{\bar{L}}+1\right).$$

The average service time is largely varied from $\bar{L} = P_{\max}$ to $\bar{L} = P_{\max}+1$ and the kink is induced by this variation. The calculation results are in good agreements with measurement results without a range over 10000 bytes. The model well explains these systems below 10000 bytes. In a range over 10000 bytes, the result is not in agreement. Our model has no limit of message lengths but an actual DCP system has a limit message length to serve at one time. In the range, the length frequently exceeds the limited value. In this measurement, the limit value is 22260 bytes. When a message

length is longer than the limit value, messages exceeded the limit are left and those are serviced at the next token arrival. Therefore, the measured result is longer than calculated result. The difference is not a problem when we design a system because, in this case, a required buffer size is too large to design a system, then we should not design a system used in the range.

Here, we would consider about the effect of the number of token-delay points. Recall equation (5.28) that is an average message length, and we rewrite the equation with equation (5.55), thus

$$L = \frac{\lambda\left\{s_{Line}\left[N\bar{L}_{oh} + D(P_{\max} - P_{\min})\right] + \bar{F}\right\}}{1 - \lambda s_{Line} N} \quad ,(P_{\max} \le L).$$

This equation shows that the part of $s_{Line} D(P_{\max} - P_{\min})$ plays as the average FTRTT. Therefore, if the number of token-delay points increases by one, the average increases by $s_{Line}(P_{\max} - P_{\min})$. In this system, this value is 116.32 μsec. This effect is larger than an effect of differences of average FTRTT's from the single HUB system. Therefore, it is concluded that the performance of the multiple HUB system is dominantly decided by the number of token-delay points. In other words, we should design the network to minimize the number of HUB's because the number of token delay points nearly equal to the number of HUB's.

These results lead us to the conclusion that the system can be designed by the quantitative approach.

Average transfer time

We can not measure transfer times in our system by a technical problem, because a length of messages arrived to members is too short to stamp time information. However the average transfer time is one of important parameters when we design a DCP system. We pointed out that the model well explains the multiple HUB system. Therefore, we consider that an average transfer time can be calculated with the calculated result of the average message length, equations (5.49), (5.55), and (5.56) under the system parameters that are summarized in Table 5.6. Figure 5.24 shows the calculated average transfer time of type-1 system. Figure 5.28 shows the calculated average transfer time of type-2 system. Figure 5.32 shows the calculated average transfer time of type-3 system.

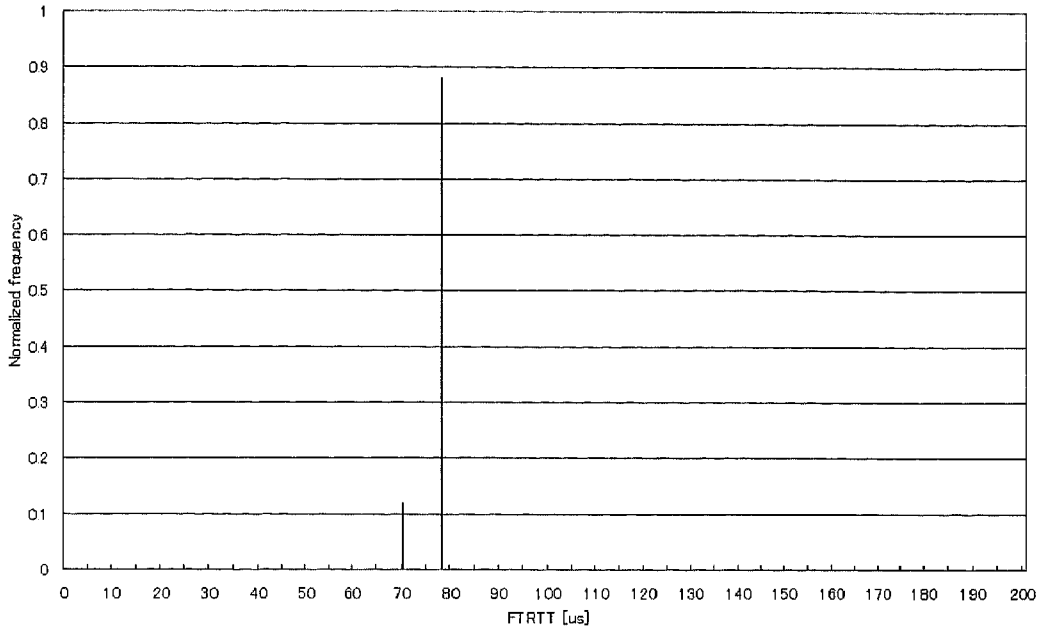

Figure 5.21: Measured FTRTT of type-1 system.

The normalized frequency is a frequency divided by the number of samples.

Table 5.7:

Measured FTRTT's of type-1 system.

Average [μsec]	Variance [μsec^2]	Number of samples
76.7	6.3	65535

Table 5.8:

Measured message lengths of type-1 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [$byte^2$]	Number of samples
5	59.6	12.6	3707.3	64872
10	160.1	22.7	26156.2	64872
15	364.5	38.1	134338.1	64872
20	1000.1	76.2	1006049.3	64872
25	2685.6	112.0	7224712.8	34173
30	12343.2	389.3	152505810.7	7204

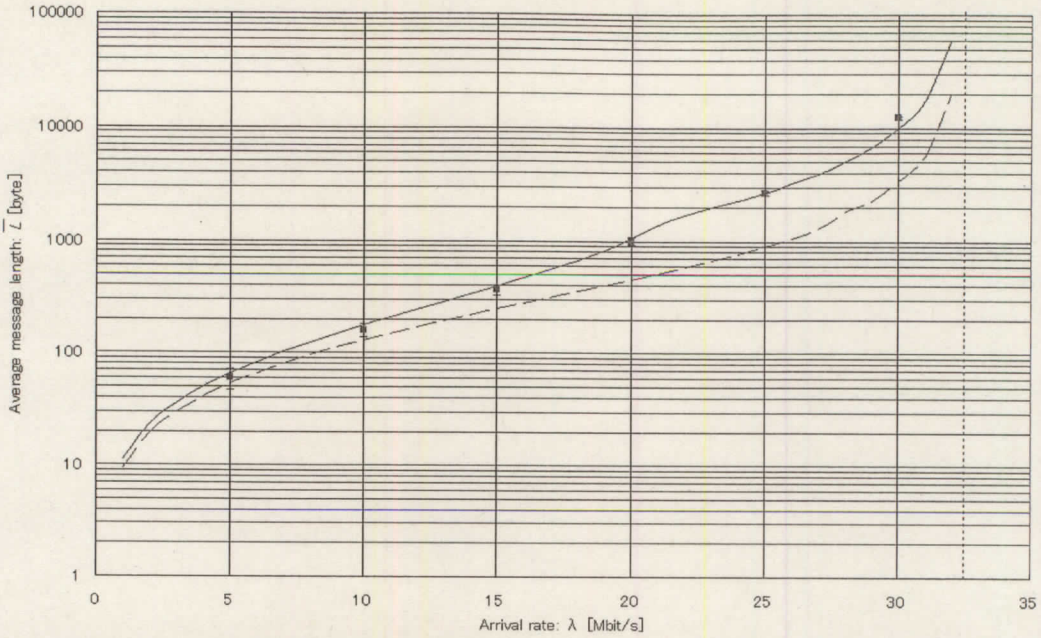

Figure 5.22: Average message length of type 1.

The solid curve displays the calculated average message length, the dashed curve displays the calculated result of the single HUB system for comparing, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

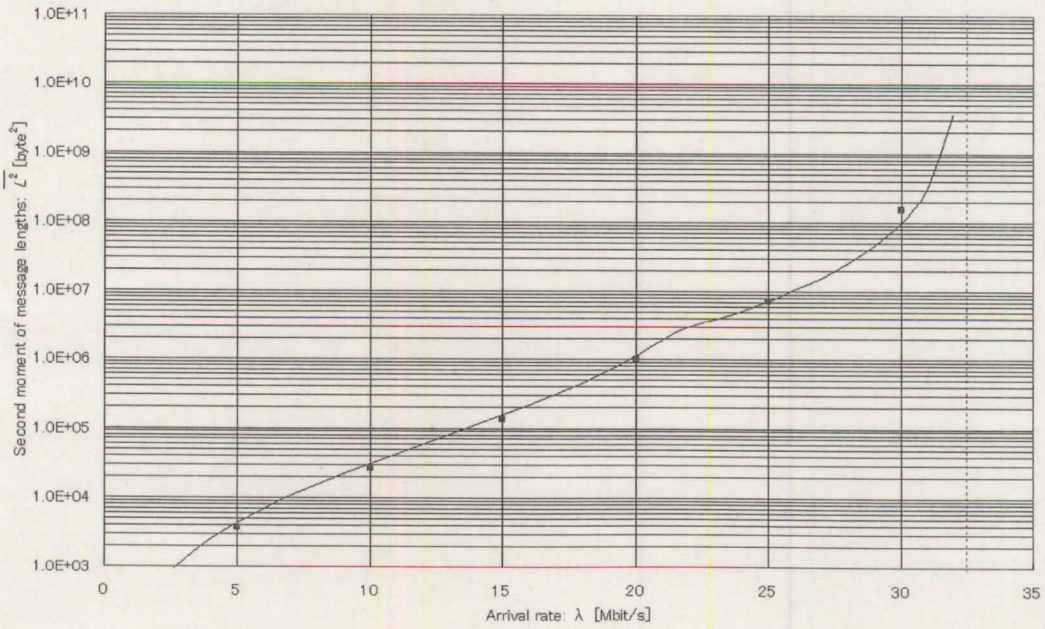

Figure 5.23: Second moment of message lengths of type-1 system.

The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

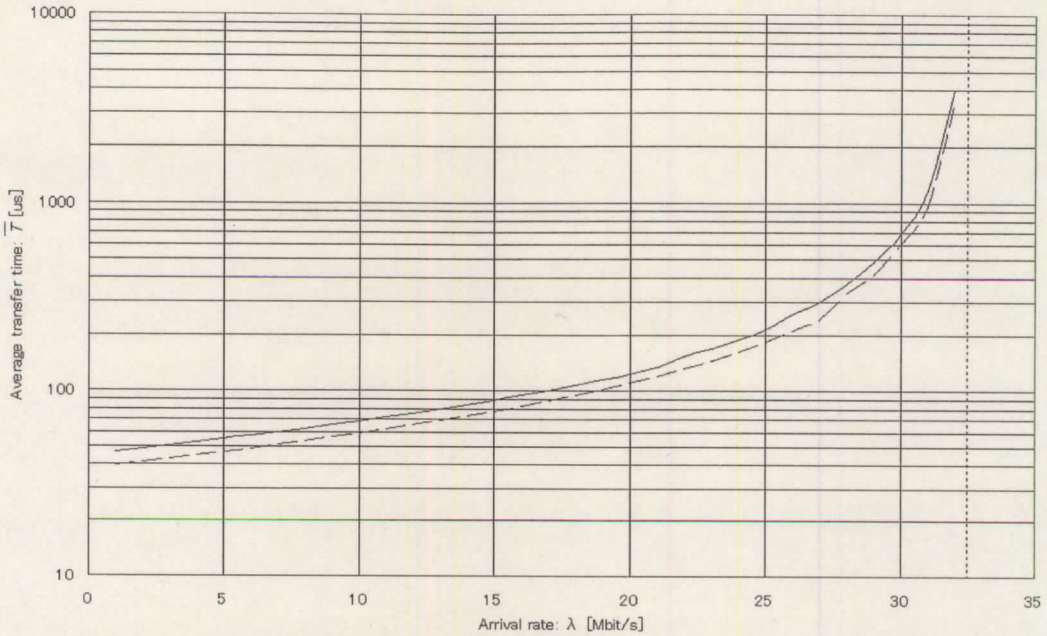

Figure 5.24: Average transfer time of type-1 system.

The solid curve displays a calculated average transfer time, the dashed curve displays the calculated result of the single HUB system for comparing, and the vertical dotted line displays the instability input rate.

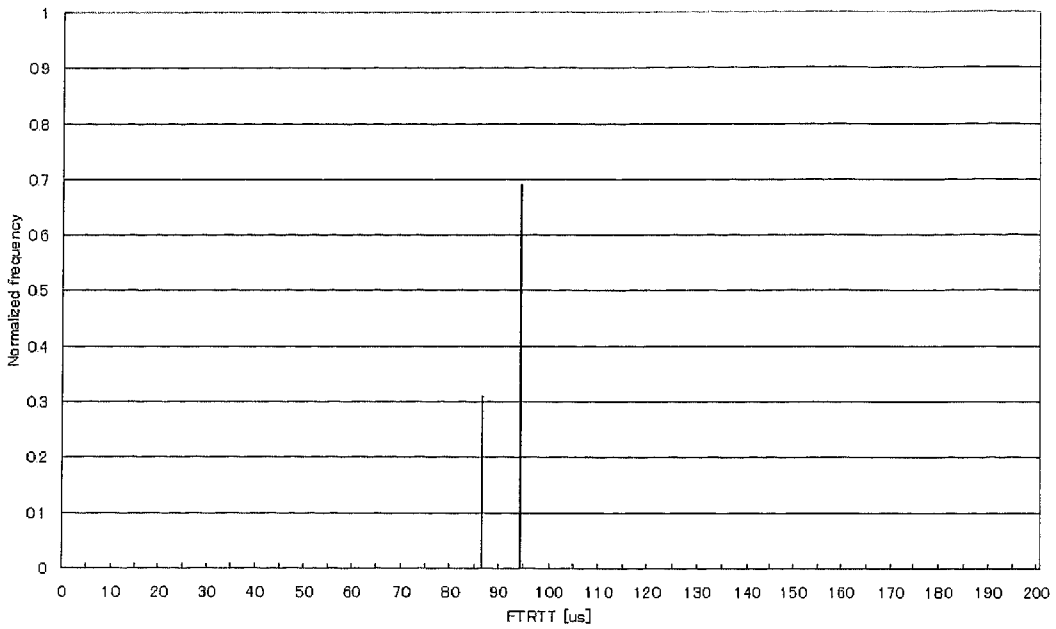

Figure 5.25: Measured FTRTT of type-2 system.

The normalized frequency is a frequency divided by the number of samples.

Table 5.9:

Measured FTRTT's of type-2 system.

Average [μsec]	Variance [μsec^2]	Number of samples
90.7	12.9	65535

Table 5.10:

Measured message lengths of type-2 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [$byte^2$]	Second moment [$byte^2$]	Number of samples
5	73.7	14.0	5625.6	64872
10	223.1	27.5	50528.2	64872
15	684.9	59.3	472565.3	64872
20	2159.0	89.1	4669396.0	34317
25	4715.7	151.9	22260434.2	16998
30	20856.8	524.5	435279981.0	4142

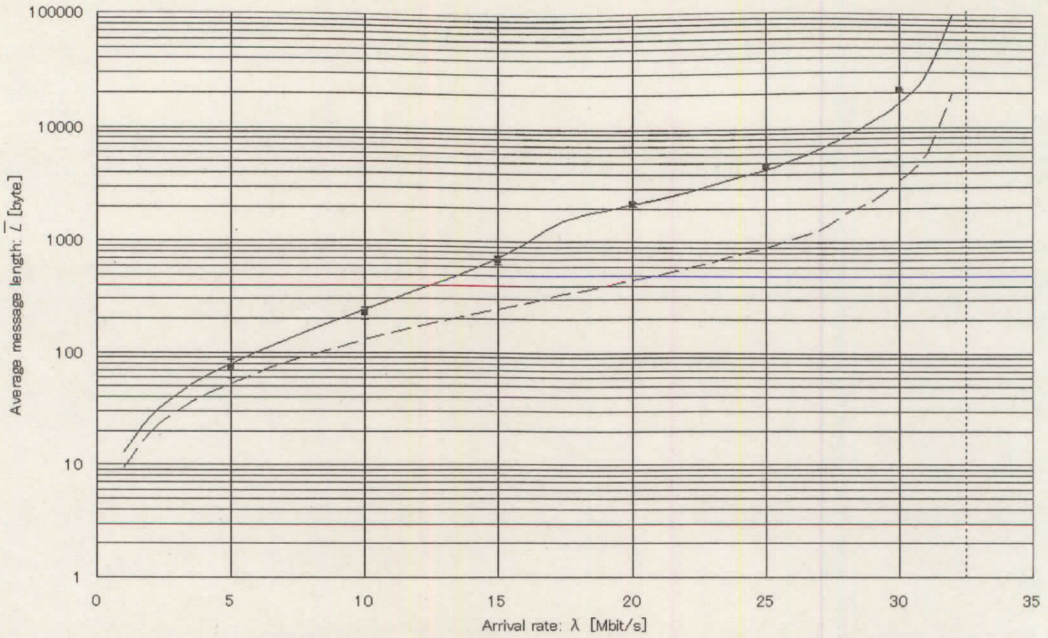

Figure 5.26: Average message length of type-2.

The solid curve displays the calculated average message length, the dashed curve displays the calculated result of the single HUB system for comparing, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

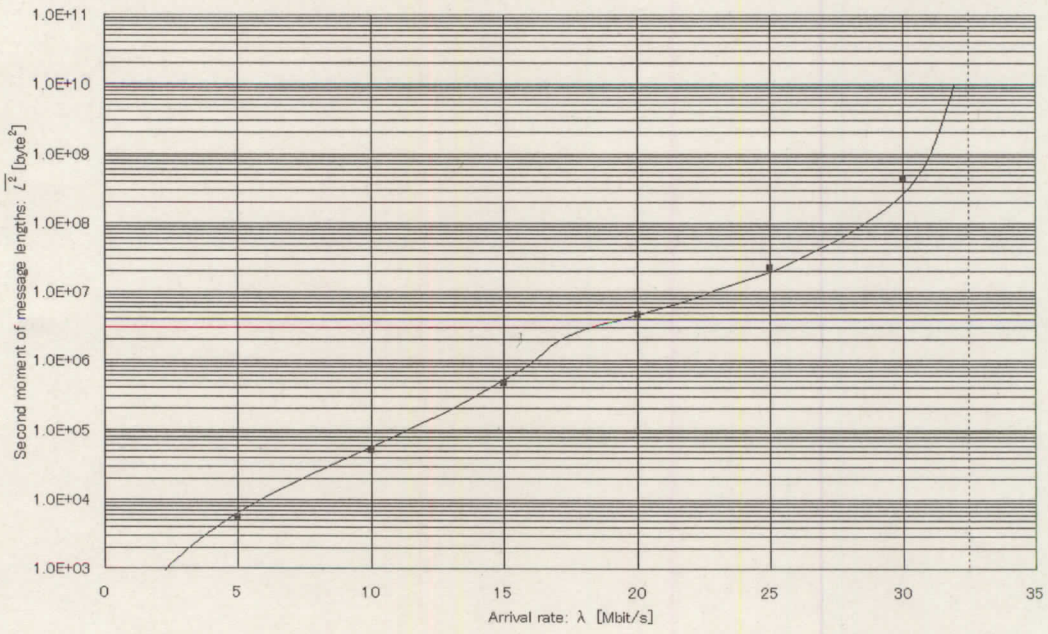

Figure 5.27: Second moment of message lengths of type-2 system.

The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

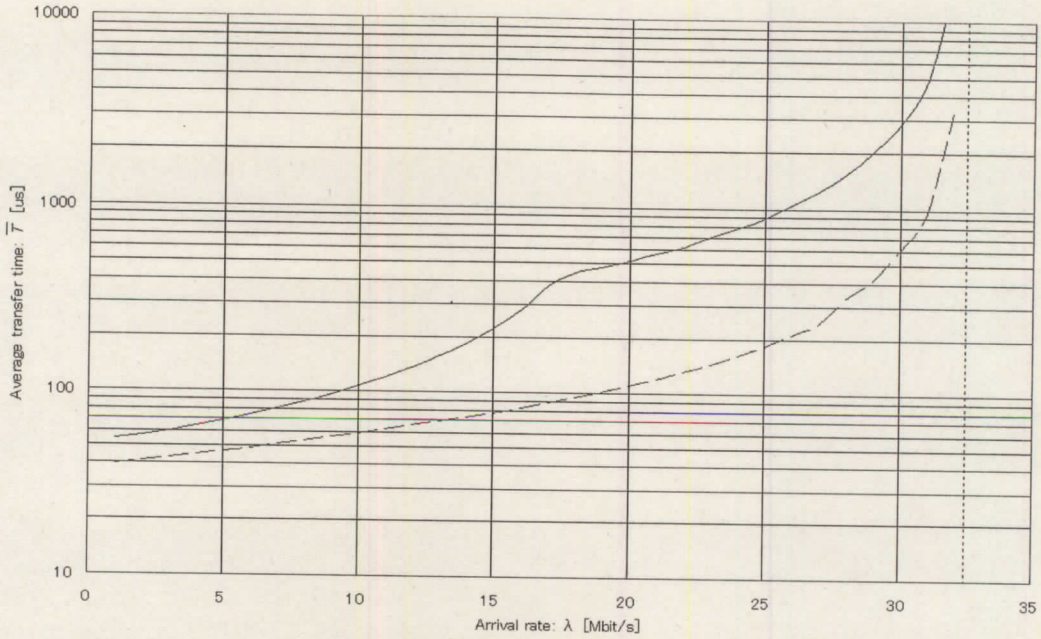

Figure 5.28: Average transfer time of type-2 system.

The solid curve displays a calculated average transfer time, the dashed curve displays the calculated result of the single HUB system for comparing, and the vertical dotted line displays the instability input rate.

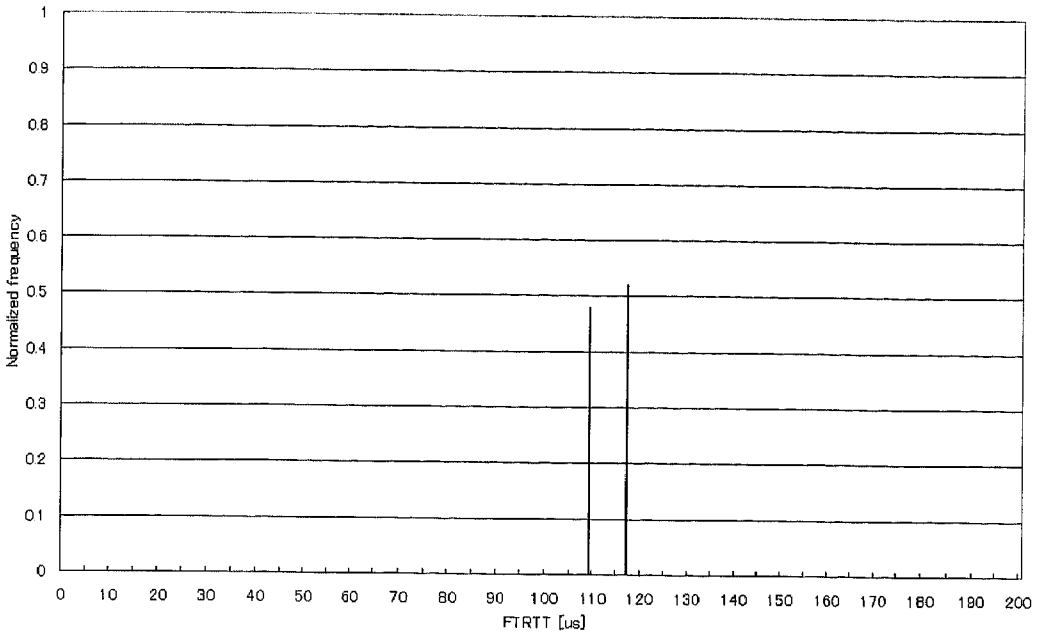

Figure 5.29: Measured FTRTT of type 3.

The normalized frequency is a frequency divided by the number of samples.

Table 5.11:

Measured FTRTT's of type-3 system.

Average [μsec]	Variance [μsec^2]	Number of samples
104.9	15.0	65535

Table 5.12:

Measured message lengths of type-3 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [$byte^2$]	Number of samples
5	89.8	15.6	8301.3	64872
10	318.4	34.1	102516.8	64872
15	1621.7	73.9	2635349.2	34822
20	3064.6	110.0	9403468.4	24282
25	6589.6	175.3	43452975.4	13332

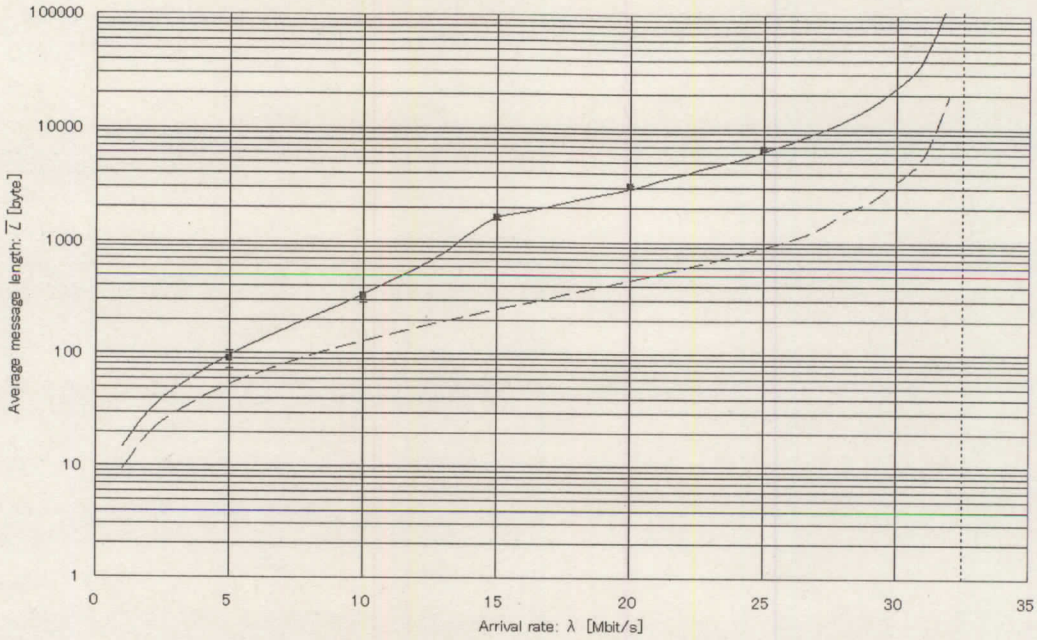

Figure 5.30: Average message length of type-3.

The solid curve displays the calculated average message length, the dashed curve displays the calculated result of the single HUB system for comparing, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

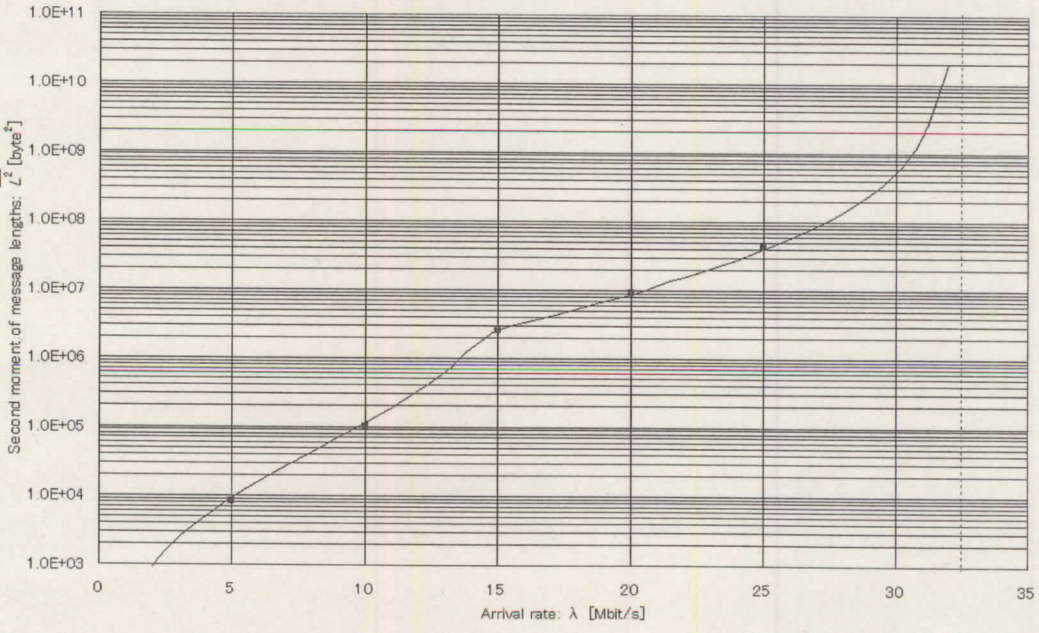

Figure 5.31: Second moment of message lengths of type-3 system.

The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

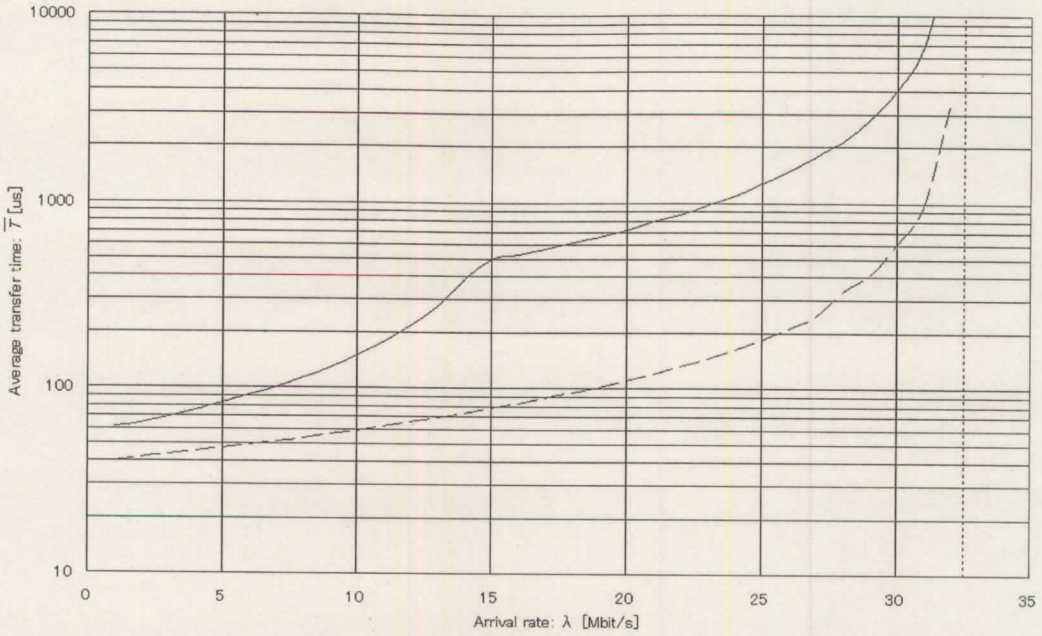

Figure 5.32: Average transfer time of type-3 system.

The solid curve displays a calculated average transfer time, the dashed curve displays the calculated result of the single HUB system for comparing, and the vertical dotted line displays the instability input rate.

5.2.4 Scalability

In this subsection, we discuss scalability of DCP. It is the scalability that the system can be constructed without a dependence on its size. In our model, the size is represented in the equations by the average FTRTT, the number of token-delay points, and the number of members. The average FTRTT and the number of token-delay points are verified in earlier subsections. One parameter has not verified yet, which is the number of members. We analyze system performance with variation of the parameter.

5.4.1 Overview

Figure 5.33 shows a test bed system that consists of six members, a collector, and three HUB's.

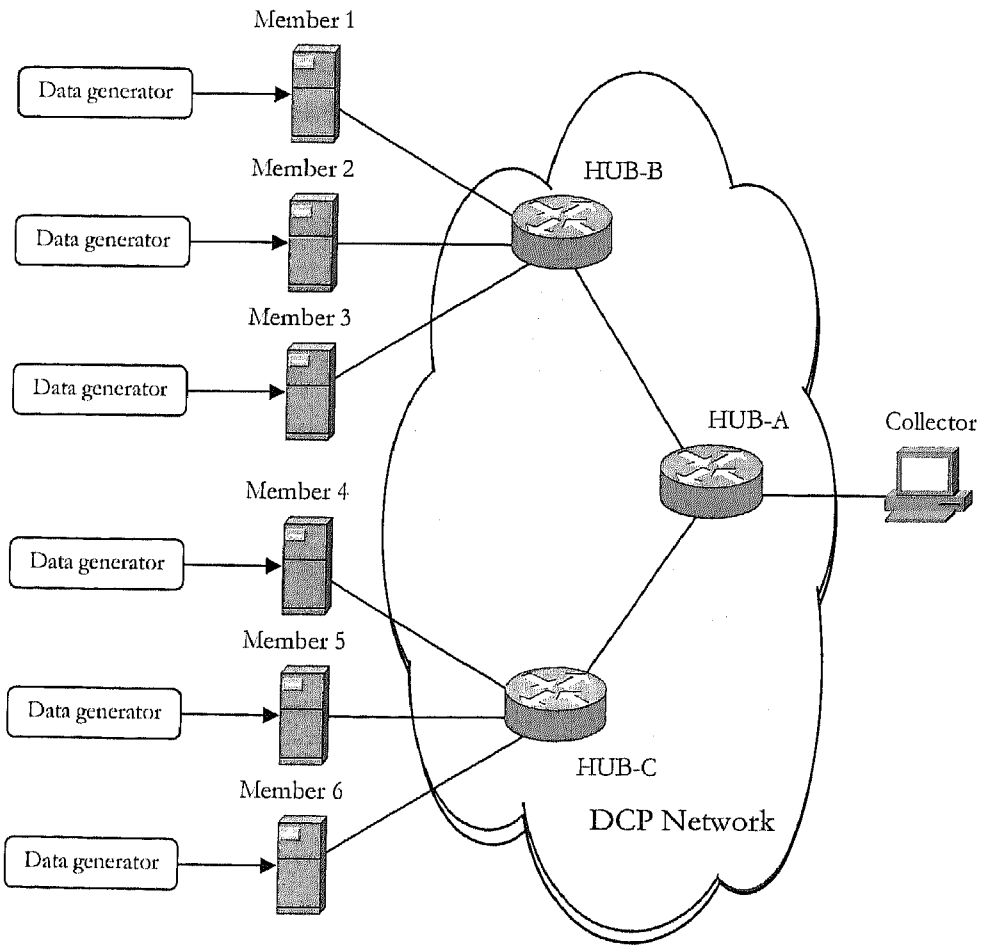

Figure 5.33: Test bed for scalability test.

This network topology is equivalent to that of type 2 in previous subsection and there are two token-delay points. In this measurement, the data generators generate Poisson arrival processes and all arrival rates are λ. A data generator generates data and the data are sent to a member. The member transmits the data to the collector via HUB's. The HUB specifications are the same in the previous subsection, which are shown in Table 5.5. We measure three systems: a system that is named X4 system consists of four members (1-4), a system that is named X5 system consists of five members (1-5), and a system that is named X6 system consists of five members (1-6). In this subsection, we discuss the following topics:

- Average FTRTT,
- Average message length,
- Second moment of message lengths, and
- Calculated average transfer time.

FTRTT

Figure 5.34 shows the measurement results of X4 system and these statistical values are summarized in Table 5.14. There are two values which are 109 and 117 μs, and other values do not exist. Figure 5.38 shows the measurement results of X5 system and these statistical values are summarized in Table 5.16. There are two values which are 132 and 140 μs, and other values do not exist. Figure 5.42 shows the measurement results of X6 system and these statistical values are summarized in Table 5.18. There are three values which are 148, 156, and 163 μs, and other values do not exist. These values are discrete and the intervals between the values are 8 or 7 μs. These discrete values do not impact our calculation because the interval of these two values is too smaller than the average value.

System parameters

We summarize system parameters of the system under measuring in Table 5.13. When we calculate a system parameter, we use the following values:

Table 5.13:

System parameters.

Parameter	Value		
	X4	X5	X6
Number of members: N	4	5	6
Number of token delay points: D	2	2	2
Transmitting period of a bit to a line: s_{Line}	10 ns	10 ns	10 ns
Maximum DCP payload length: $P_{\max}$	11872 bits (1484 bytes)	11872 bits (1484 bytes)	11872 bits (1484 bytes)
Minimum DCP payload length: $P_{\min}$	240 bits (30 bytes)	240 bits (30 bytes)	240 bits (30 bytes)
Average FTRTT: $\overline{f}$	112.6 μs	133.6 μs	154.6 μs
Variance of FTRTT's: σ_u^2	15.0 μs^2	10.3 μs^2	4.8 μs^2
Header length: L_{hd}	336 bits	336 bits	336 bits

Average and second moment of message lengths

We measured the average message length at member 1. We calculate the average message length and the second moment of message lengths based on the parameters listed in Table 5.13. The average message length is calculated with the equations (5.47), (5.55) and (5.58). The second moment of message lengths is calculated with the calculated result of the average message length, equations (5.56) and (5.48).

The measured message lengths of X4 system are summarized in Table 5.15. The average message length of measured values and calculated results are displayed in Figure 5.35. In the similar fashion, shows the second moment displayed in Figure 5.34. The measured message lengths of X5 system are summarized in Table 5.17. The average message length of measured values and calculated results are displayed in Figure 5.39. In the similar fashion, shows the second moment displayed in Figure 5.38. The measured message lengths of X6 system are summarized in Table 5.19. The average message length of measured values and calculated results are displayed in

Figure 5.42. In the similar fashion, shows the second moment displayed in Figure 5.41. In these figures the solid curve, and closed squares display the calculated results, and measured values, respectively. And the dotted vertical line displays the instability arrival rate that is calculated by the stability condition equation (5.45), definition (5.31), $\bar{s} = 10\,\text{ns}$, and N=4, 5, or 6. We find a kink of the calculated result at L=1484 bytes in the figures. Its reason comes from a large variation of the average service time, which is the same reason in a multiple HUB system. The calculation results are in good agreements with measurement results without a range over 10000 bytes. The reason is the same of a multiple HUB system. The differences are not problems when we design a system, which is discussed in the previous subsection.

These results lead us to the conclusion that the system can be designed by the quantitative approach.

Average transfer time

We can not measure transfer times in our system by technical problems because a length of messages arrived to members is too short to stamp time information. However the average transfer time is one of important parameters when we design a DCP system. We pointed out that the model well explains the system. Therefore, we consider that the average transfer time can be calculated with the calculated result of the average message lengths, the equations (5.55), and (5.56) based on the parameters listed in Table 5.13. Figure 5.37 shows the calculated average transfer time of X4 system. Figure 5.41 shows the calculated average transfer time of X5 system. Figure 5.45 shows the calculated average transfer time of X6 system. In these figures the solid curve displays the calculated result. And the dotted vertical line displays the instability arrival rate that is calculated by the equations (5.45), (5.31), $\bar{s} = 10\,\text{ns}$, and N=4, 5, or 6.

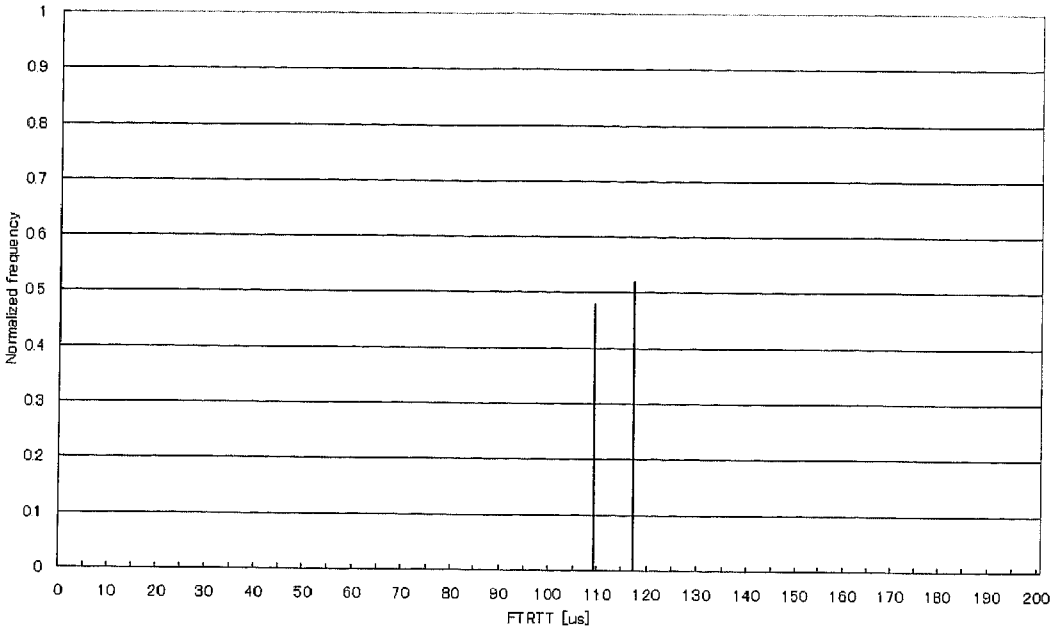

Figure 5.34: Measured FTRTT of X4 system.

The normalized frequency is a frequency divided by the number of samples.

Table 5.14:
Measured FTRTT's of X4 system.

Average [μsec]	Variance [μsec^2]	Number of samples
112.6	15.0	65535

Table 5.15:
Measured message lengths of X4 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [$byte^2$]	Number of samples
2.5	41.4	22.9	2234.7	49164
5.0	100.1	35.2	11247.0	48713
7.5	190.8	34.8	37625.2	48240
10.0	353.4	35.5	126139.1	47868
12.5	719.0	58.9	520269.1	47466
15.0	1757.3	80.4	3094576.5	24488
17.5	2771.3	109.2	7691730.3	24016
20.0	5295.4	173.6	28070979.5	11987
22.5	16713.7	585.8	279689440.1	3739

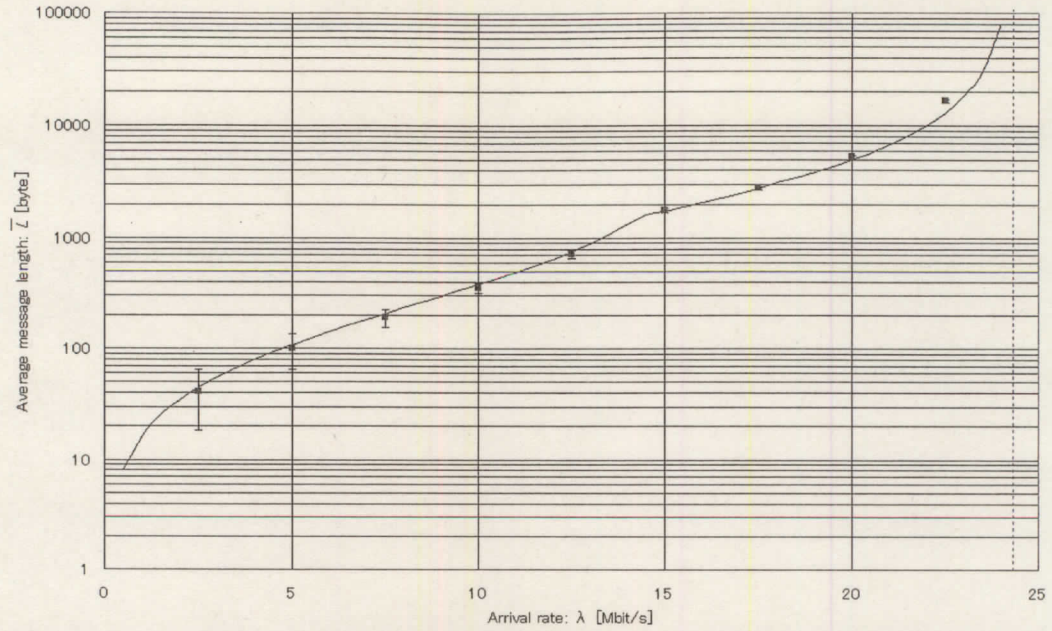

Figure 5.35: Average message length of X4 system.

The solid curve displays the calculated average message length, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

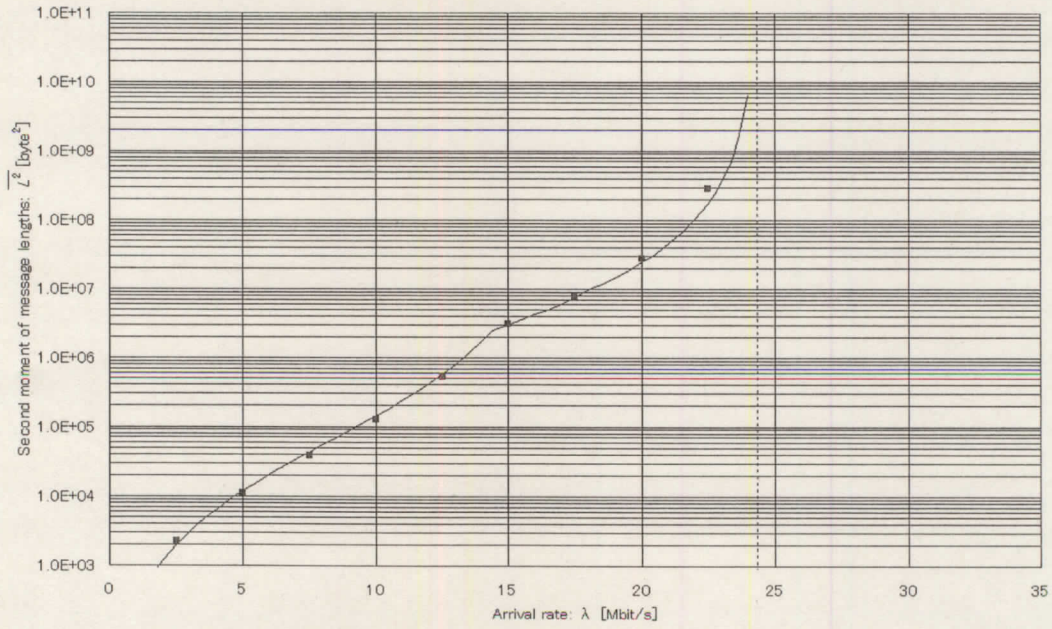

Figure 5.36: Second moment of message lengths of X4 system.

The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

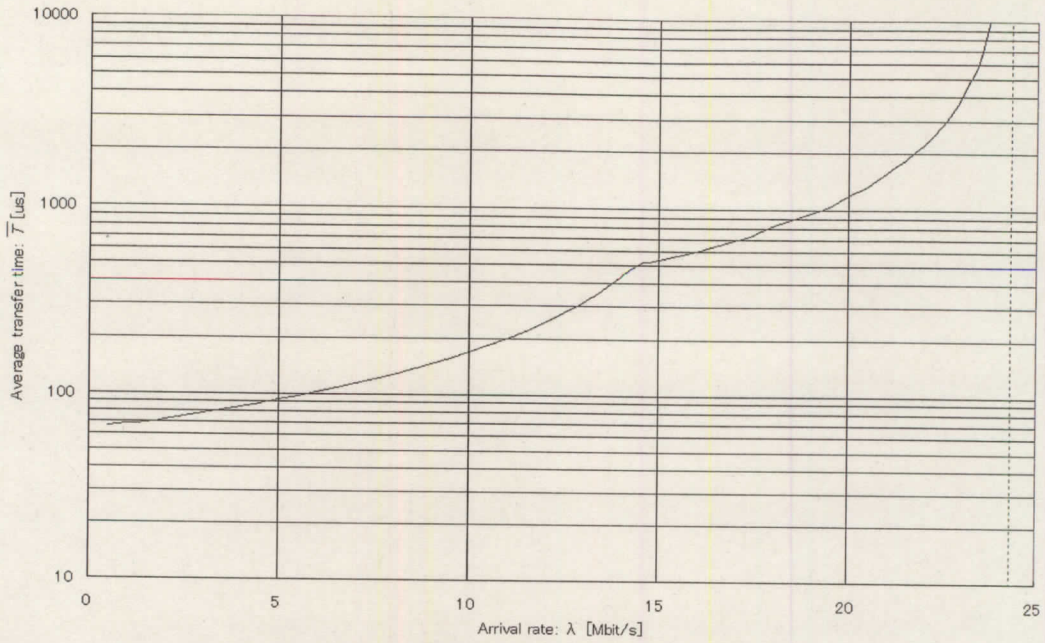

Figure 5.37: Average transfer time of X4 system.

The solid curve displays a calculated average transfer time, and the vertical dotted line displays the instability input rate.

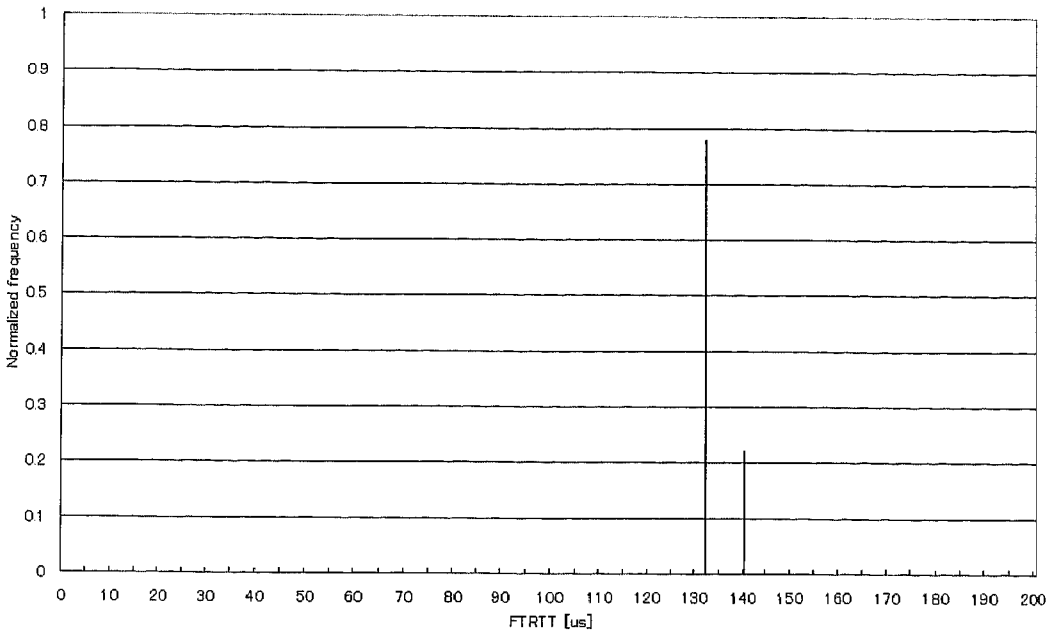

Figure 5.38: Measured FTRTT of X5 system.

The normalized frequency is a frequency divided by the number of samples.

Table 5.16:

Measured FTRTT's of X5 system.

Average [μsec]	Variance [μsec^2]	Number of samples
133.6	10.3	65535

Table 5.17:

Measured message lengths of X5 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [$byte^2$]	Number of samples
2.5	50.7	24.0	3147.4	49228
5.0	129.4	55.5	19820.3	48739
7.5	264.5	27.9	70707.3	48270
10.0	569.1	47.2	326105.0	47930
12.5	1673.2	79.2	2805705.0	24669
15.0	3237.9	125.6	10499702.3	16339
17.5	9689.9	310.0	93990160.3	6724

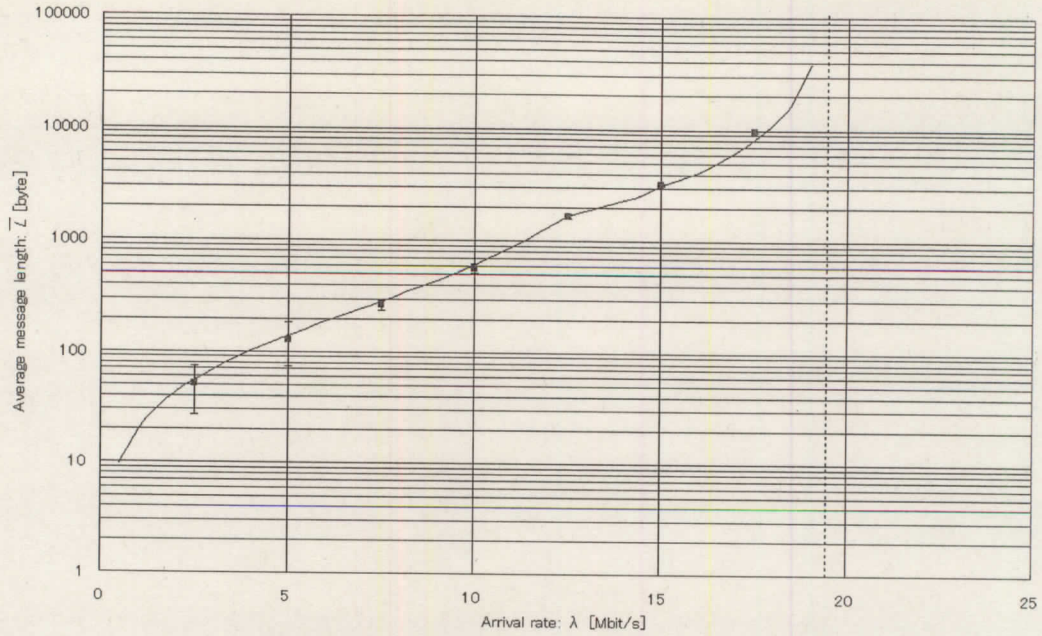

Figure 5.39: Average message length of X5 system.
The solid curve displays the calculated average message length, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

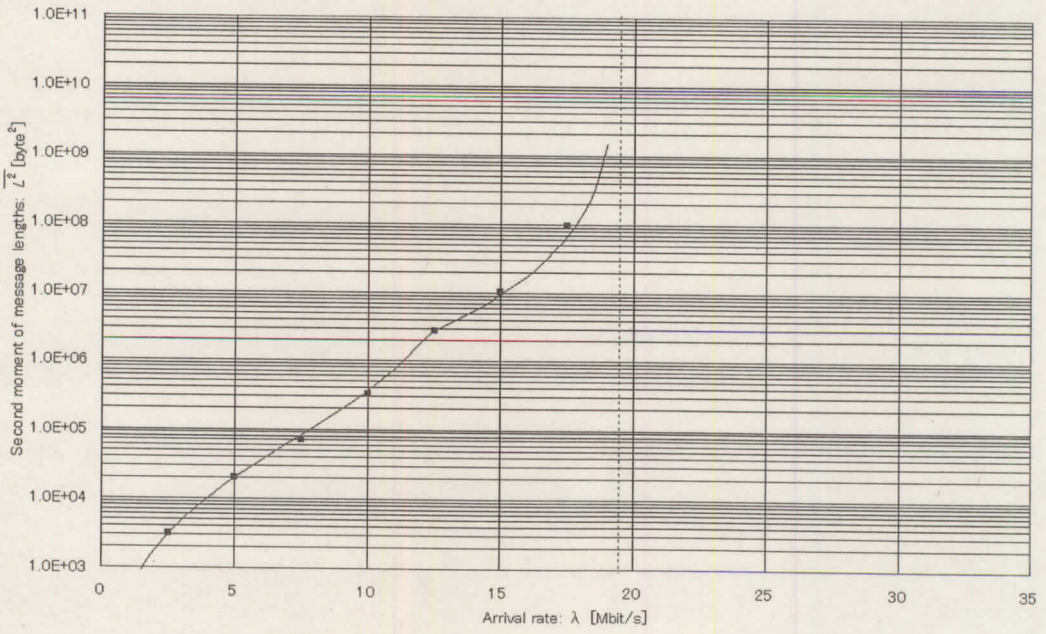

Figure 5.40: Second moment of message lengths of X5 system.
The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

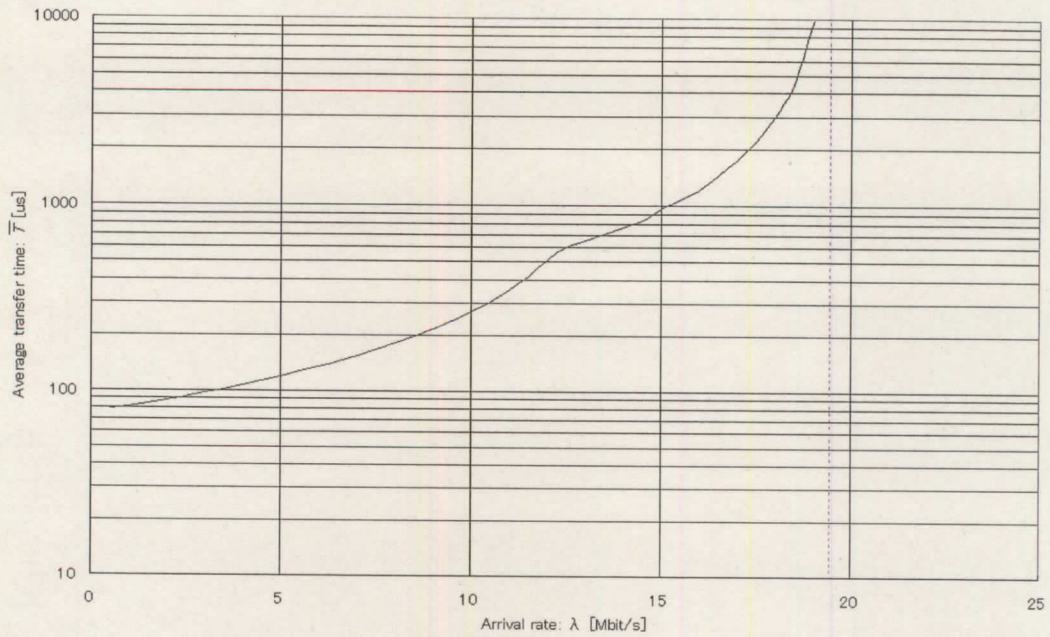

Figure 5.41: Average transfer time of X5 system.

The solid curve displays a calculated average transfer time, and the vertical dotted line displays the instability input rate.

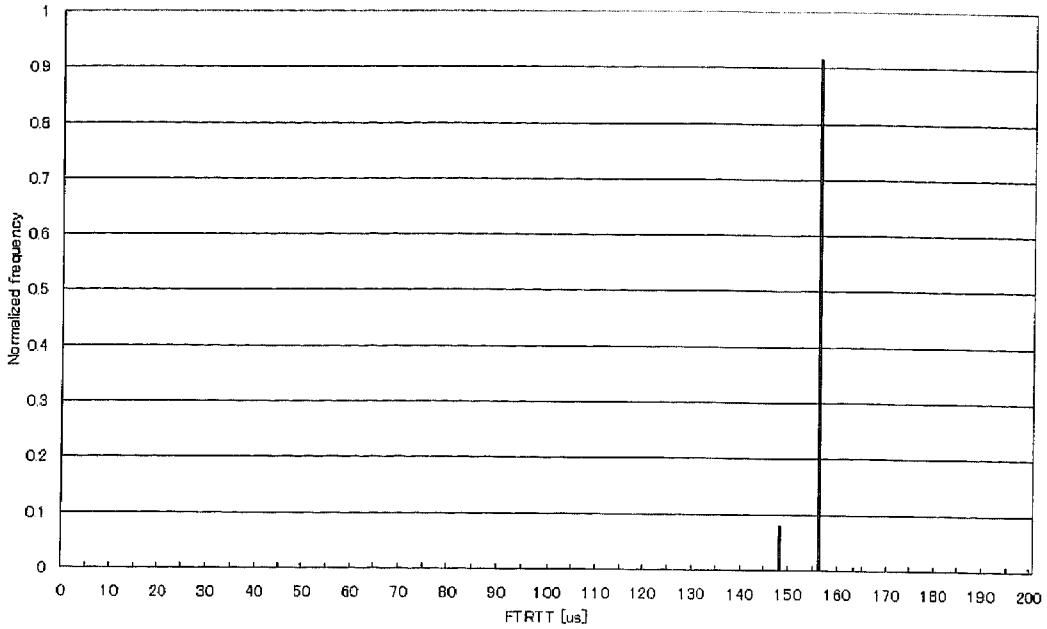

Figure 5.42: Measured FTRTT of X6 system.

The normalized frequency is a frequency divided by the number of samples.

Table 5.18:

Measured FTRTT's of X6 system.

Average [μsec]	Variance [μsec^2]	Number of samples
154.6	4.8	65535

Table 5.19:

Measured message lengths of X6 system.

Arrival rate λ [Mbit/s]	Average [byte]	Standard deviation [byte]	Second moment [$byte^2$]	Number of samples
2.5	60.8	25.2	4330.4	49327
5.0	162.6	45.3	28478.4	48843
7.5	366.9	34.0	134421.1	48359
10.0	1016.3	70.0	1037660.0	47982
12.5	2723.6	115.8	7431116.0	24338
15.0	12830.6	437.0	164815038.0	5146

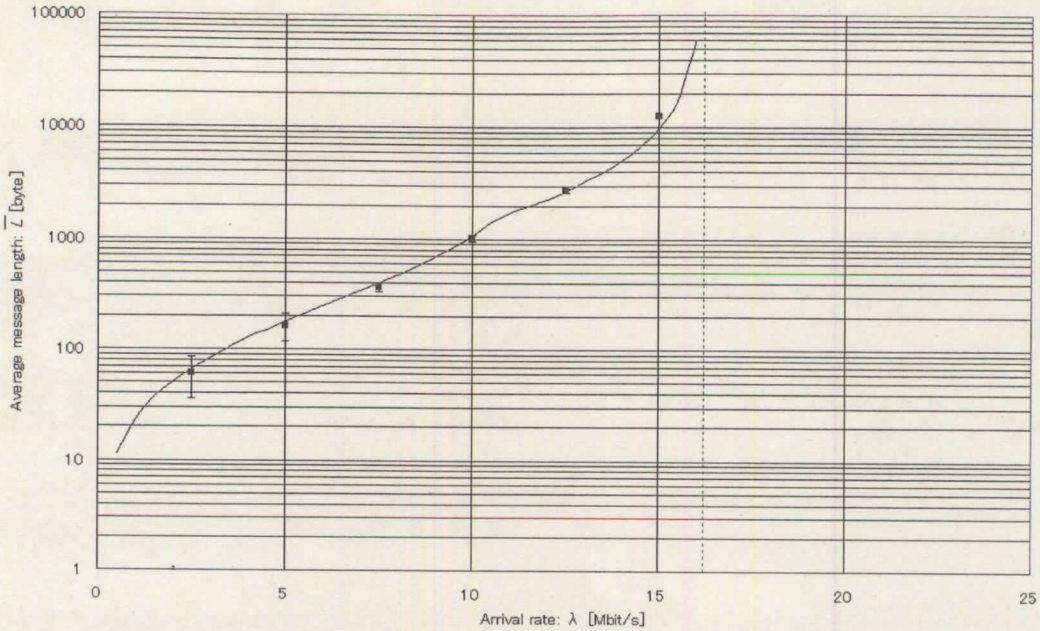

Figure 5.43: Average message length of X6 system.

The solid curve displays the calculated average message length, closed squares display the data points together with the associated error bars, and the vertical dotted line displays the instability input rate.

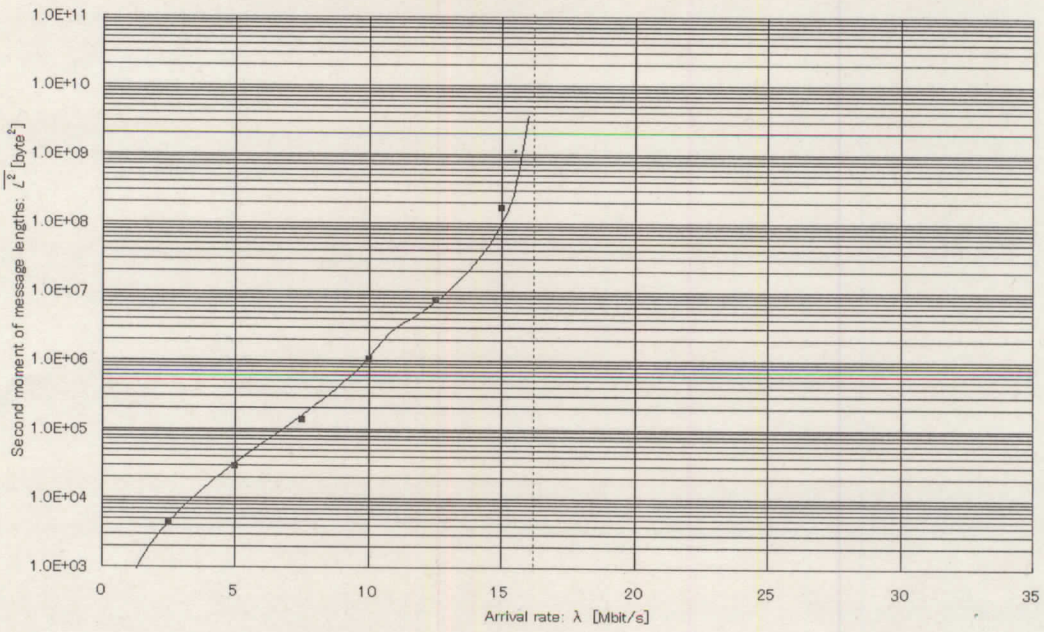

Figure 5.44: Second moment of message lengths of X6 system.

The solid curve displays the calculated average message length, closed squares display the data points, and the vertical dotted line displays the instability input rate.

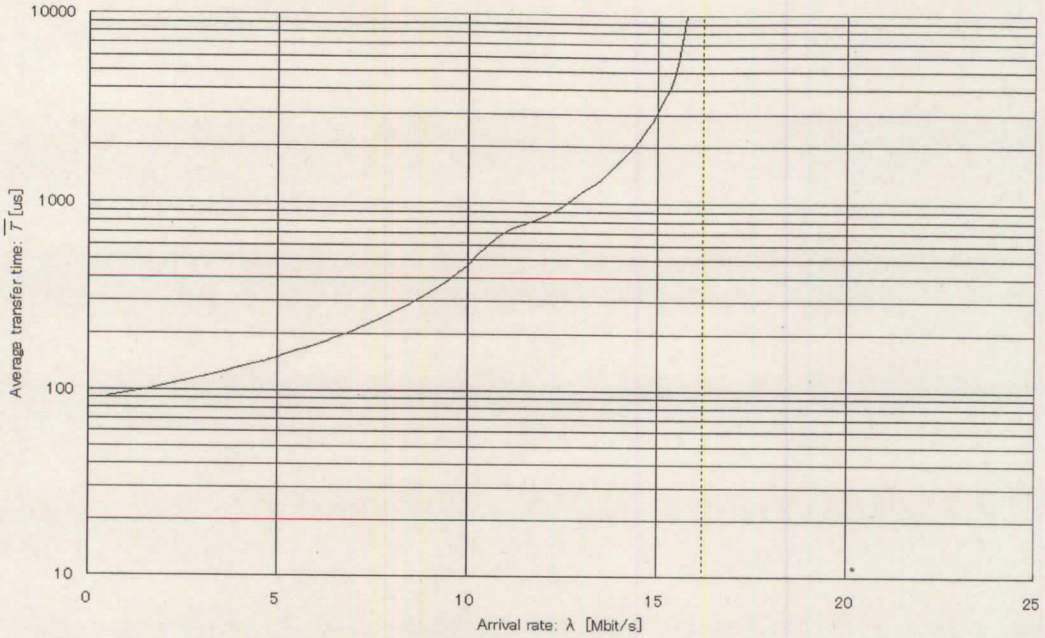

Figure 5.45: Average transfer time of X6 system.

The solid curve displays a calculated average transfer time, and the vertical dotted line displays the instability input rate.

5.3 Comparison of DCP and TCP

In this section, we compare DCP and TCP in terms of their performance. In order to compare them, we measured data transfer rate variations at a receiver when many senders transmit data to the receiver. In the first subsection, we discuss the transfer variations of a DCP system. In the second subsection, we discuss the variation of a TCP system. In the last subsection, we compare those results and point out the DCP system is more suitable for the DAQ system than the TCP system.

5.3.1 DCP

The test bed of this measurement is shown in Figure 5.46, which consists of three members, a collector, and a HUB. The member is implemented by hardware and the collector is implemented by software. This system is the identical system that is measured and analyzed in section 5.2 but the manner of data generation of data generator in members is different. Each member transfers data in a size 100 Mbytes to the collector as fast as possible and these data aggregate at the HUB. We used two different HUB's, these specifications are summarized as Table 5.20. There is not a large difference.

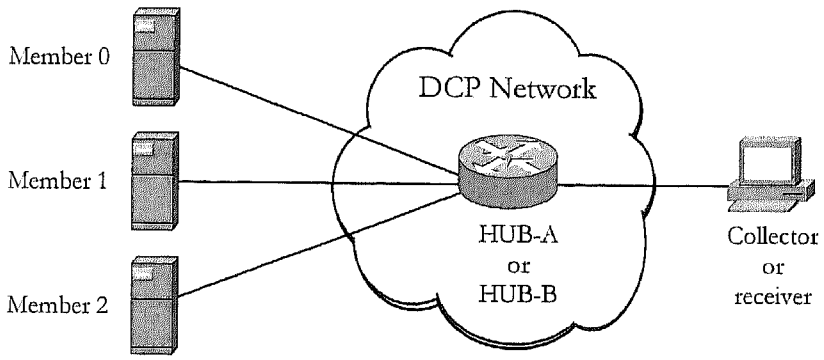

Figure 5.46: Test bed for the DCP measurement.

Table 5.20:
Summary of HUB specifications.

	HUB-A	HUB-B
Model	BUFFALO LSW10/100-5P	COREGA FSW-8A
Frame switching mechanism	Store and forward	Store and forward
Max forwarding throughput	148810pps (100BASE-T)	148800pps (100BASE-T)
Number of port	5 (RJ45)	8 (RJ45)
MAC address table size	4096	2000
Packet buffer size (byte)	128K	128K
Physical characteristics (mm)	105(W) x 79(D) x 26(H)	177(W) x 133(D) x 31(H)
Weight (g)	145	330

The results are shown in Figures 5.47 and 5.48. We can find that transfer rate variations are decreases extremely every about 1 s. The decreases are due to a kernel of Linux OS and then this behavior is not observed at the FPGA card.

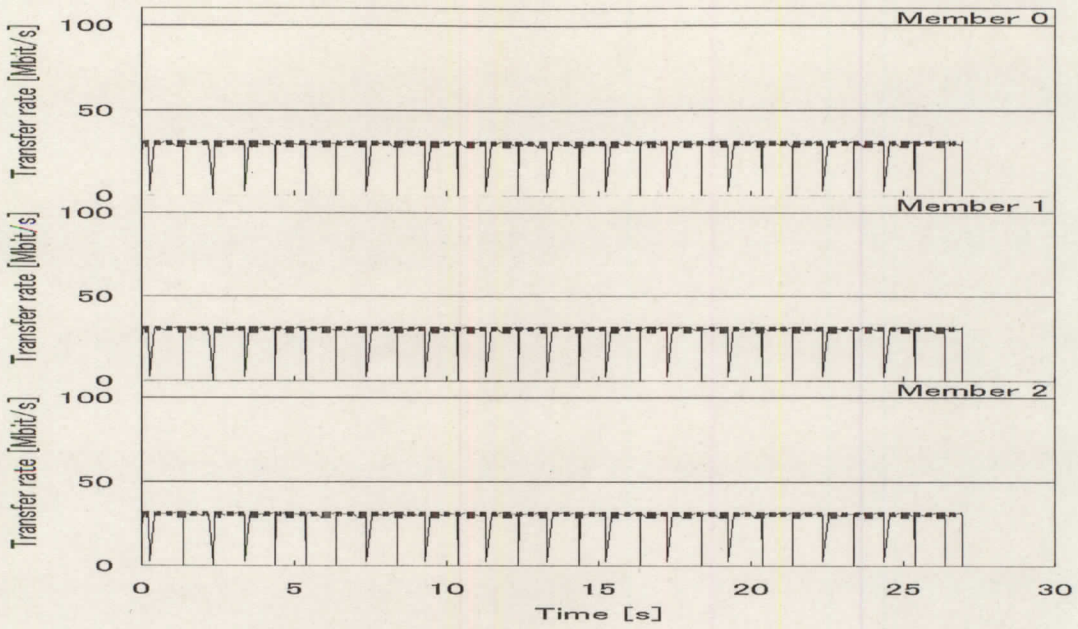

Figure 5.47: DCP transfer rate variation with HUB-A.

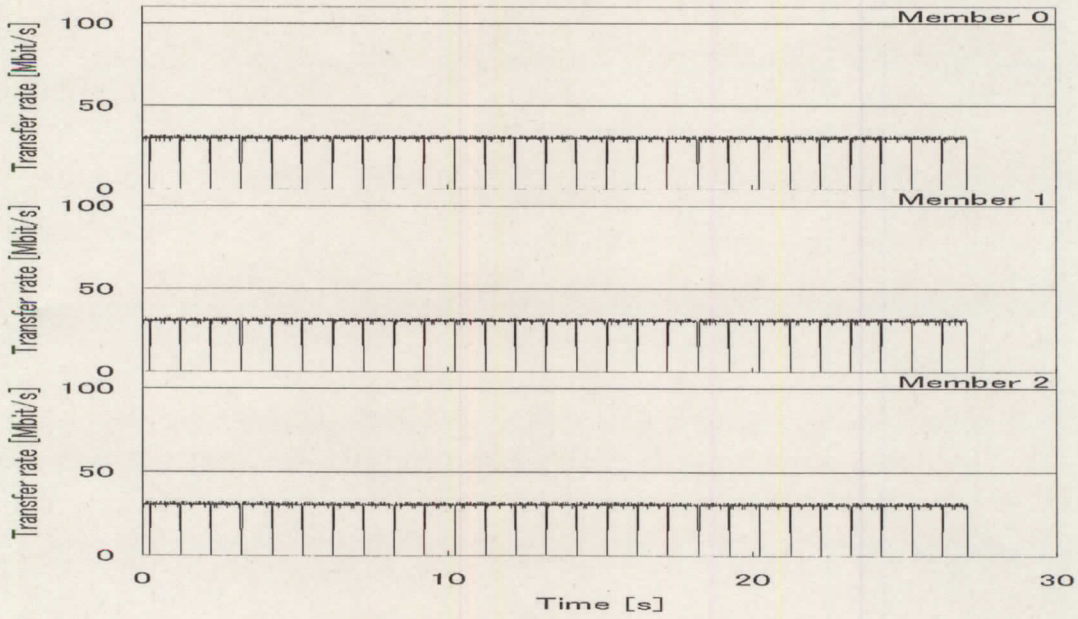

Figure 5.48: DCP transfer rate variation with HUB-B.

5.3.2 TCP

The test bed of this measurement is shown in Figure 5.49, which consists of three senders, a receiver, and a HUB. The senders and a receiver are PC's. These PC specifications are summarized in Table 5.21. We make up programs based on a Linux OS for the measurement with *socket* functions.

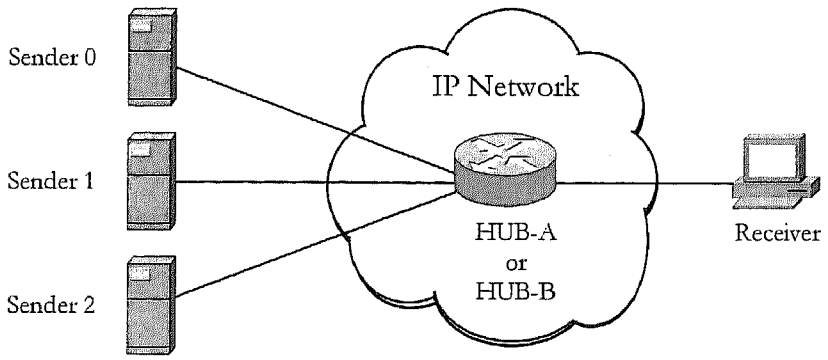

Figure 5.49: Test bed for the TCP measurement.

Table 5.21:
Specifications of PC's

	CPU	RAM	Linux kernel version
Sender 0	Celeron 1.3 GHz	128 Mbyte	2.6
Sender 1	Celeron 2.6 GHz	256 Mbyte	2.6
Sender 2	Celeron 2.0 GHz	512 Mbyte	2.4
Receiver	Celeron 1.0 GHz	256 Mbyte	2.4

In order to measure under the same condition of the DCP performance measurement, each sender transfers data in a size 100 Mbytes to the receiver as fast as possible and the receiver transmits only control packets to senders.

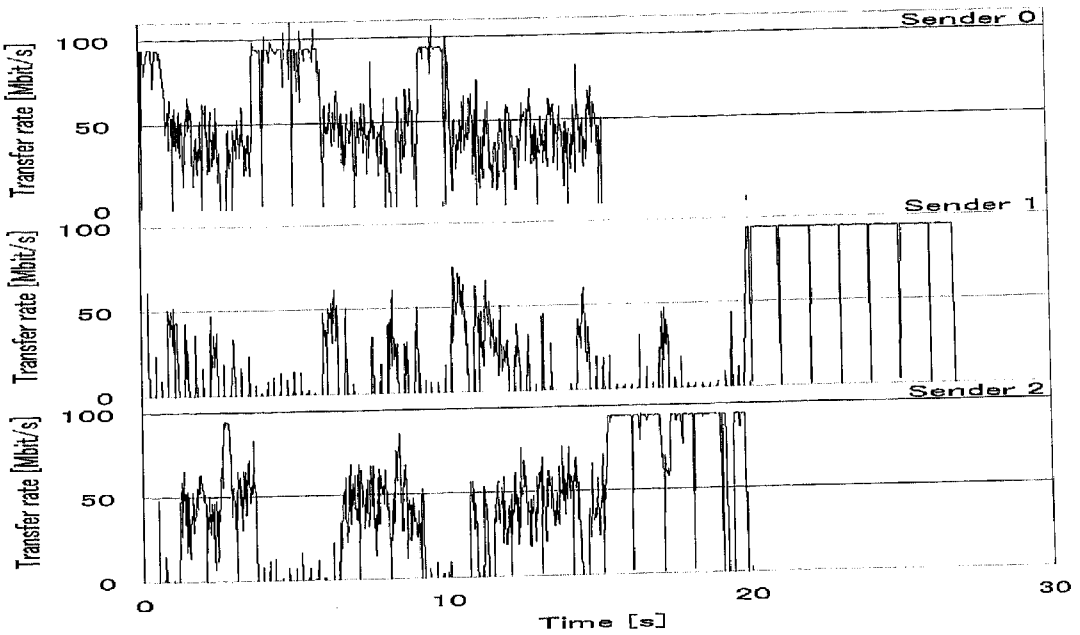

Figure 5.50: TCP transfer rate variation with HUB-A.

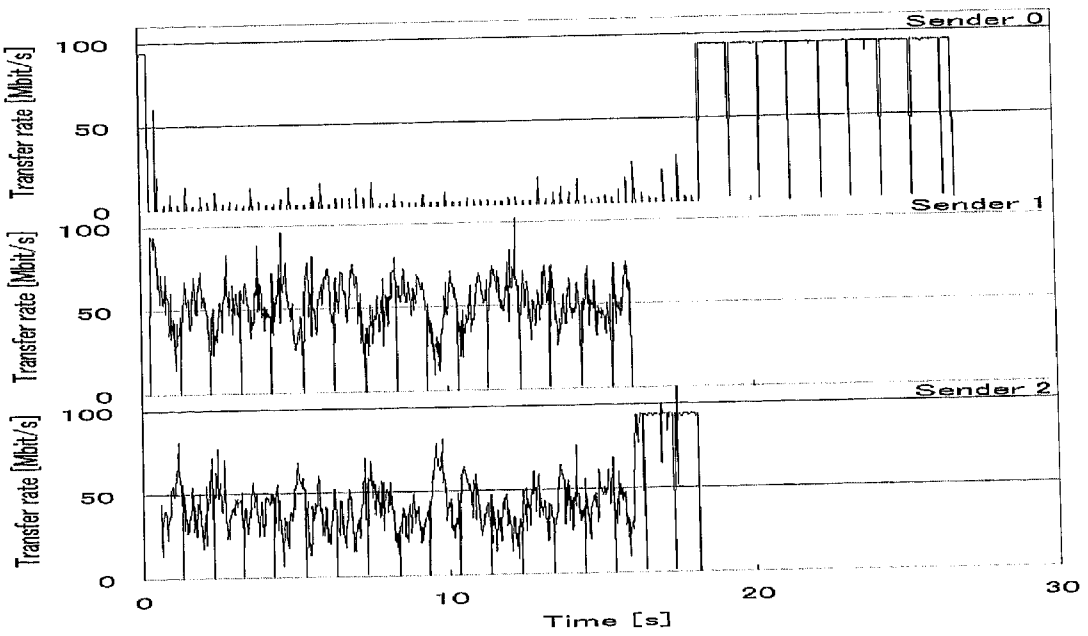

Figure 5.51: TCP transfer variation with HUB-B.

Figures 5.50 and 5.51 show results obtained with HUB-A and HUB-B, respectively. The transfer rate variations are very large and transfer behavior is different from each other. The behavior depends on HUB's and is complicated. And these results show that the TCP transfer behavior strongly depends on characteristics of network devices such as a HUB. The main reason of these complicated behaviors comes from a re-transmission mechanism of TCP. The TCP executes a best-effort transfer and all senders try to independently transmit data simultaneously. Since data can not be simultaneously transferred, the transfer induces packet losses at a HUB and re-transmissions occur in order to recover lost data. In consequence, these independent re-transmissions make complicated behavior.

5.3.3 Comparison

In the case of TCP, three active flows compete for a bandwidth. The protocol has a strong dependence on HUB's in spite of similar specification (see Table 5.20) as long as quoted by the manufacturers. This result shows that the packet switching behavior depends on internal structures of switches. It is difficult for predicting or evaluating the performance. Since TCP has a strong dependence on HUB's and the data transfer behavior is complicated, the performance prediction is not straightforward.

In contrast to TCP, data transfer behavior of DCP is quite simple. It does not have strong dependence on switches and we can predict the performance with queuing theory. Since the token passing mechanism keeps fairness transfer among senders, all transfer rate variations are to be the same.

As discussed above, we can conclude that DCP is more suitable protocol for the DAQ system than TCP.

References

[1] L. Kleinrock, "Queuing system, Volume 1: Theory", Jhon Wiley and Sons, 1975.

[2] H. Takagi, "Queuing analysis of polling models", ACM Computing Surveys, 20, 5-28, 1988.

[3] H. Takagi, "Analysis and application of polling models", Institute of Policy and Planning Sciences discussion paper series No.805, University of Tsukuba, 1998.

[4] M. Ferguson and Y. Aminetzah, "Exact Results for Nonsymmetric Token Ring Systems", IEEE Trans. Commun., 33, 223-231, 1985.

[5] O. Hashida, "Gating multiqueues served in cyclic order", Trans. Inst. Electron. Commun. Eng. Jpn. 53-A, 43-50, 1970.

[6] W. Szpankowski, "Towards computable stability criteria for some multidimensional stochastic process", Stochastic analysis of computer and Communication Systems (Ed. H. Takagi), 131-172, Elsevier Science Publishers B.V., 1985.

[7] IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE standard 802.3, 2003.

Chapter 6

Extension

In this chapter, we discuss extension of DCP. First we discuss criteria to achieve high performance in a DCP network. In the next three sections, techniques to construct a large scale DCP network system are discussed. In the last section, we introduce one of applications for a HEP DAQ system, which is COPPER (COmmon Pipelined Platform for Electronics Readout) [1] [2].

6.1 Criteria to achieve high performance in a DCP network

In this section, we discuss criteria to achieve high performance in a DCP network. Since a member has not a method to control a transmission rate of packets, a member effort to use a maximum bandwidth that is decided by their network interface specification when the member transmits data packets. Therefore, we should carefully design a network to achieve high performance. It is the criterion that an interface rate of a member is equal or smaller than all ones along the path to a collector and all members should satisfy this criterion.

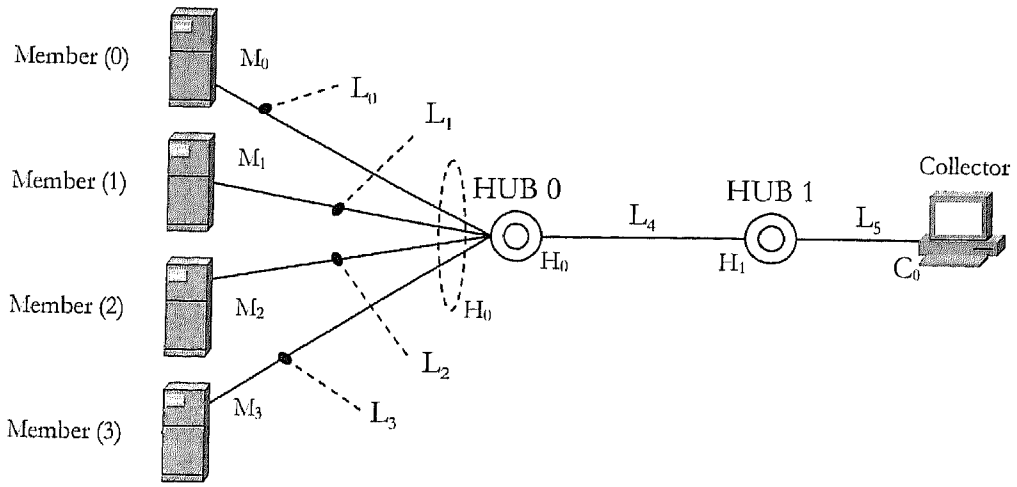

Figure 6.1: DCP system

Next, we consider the DCP system that consists of four members for example. The system under consideration is shown in Figure 6.1, where M, H and C denote interface rates and each L represent a line rate. There are four members, two HUB's and a collector. If a line is connected to different interface rates, its rate is selected a smaller one. Therefore, L_i ($0 \le i \le 5$) are

$$L_i = \min(M_i, H_0), \quad i = 0,1,2,3$$

$$L_4 = \min(H_0, H_1)$$
$$L_5 = \min(H_1, C_0).$$

We use these representation and the criteria are

$$L_i \le \min(L_4, L_5), \quad i = 0,1,2,3.$$

If those criteria are not satisfied, congestion occurs and packet losses are induced.

6.2 Multiple group system

In an event builder, a detector subsystem transmits data to two or more event processors. The multiple DCP group system is useful for this case. We would discuss an $N \times M$ event-builder that consists of N detector subsystems and M event processors. In this system, the subsystem transmits event fragment data to M event-processors.

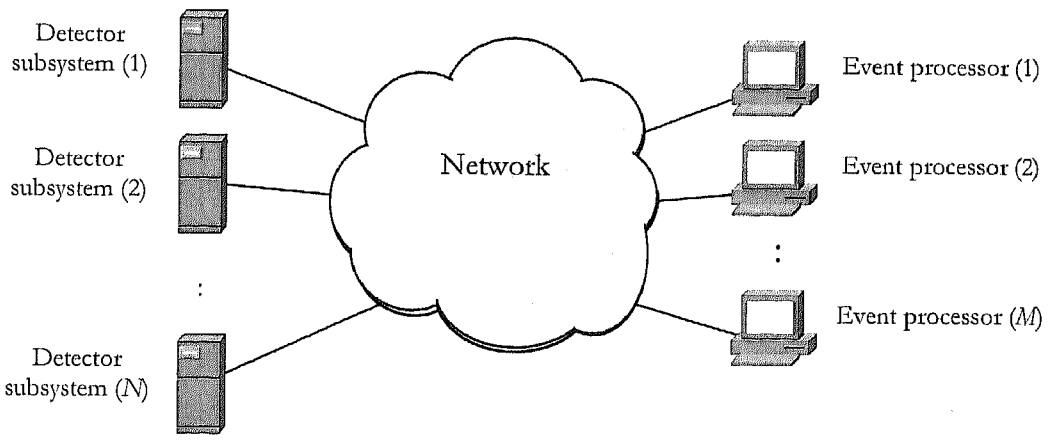

Fig. 6.2: $N \times M$ event-builder.

Figure 6.2 shows an $N \times M$ event-builder. A detector subsystem transmits an event-fragment data to a selected event processor among M event processors. The

selection rule of event processors depends on the system design, for example, the processor is selected in cyclic by the event-number of the event-fragment data.

We discuss a method of a system realization with DCP. Since a DCP token-ring is logical ring, we are abele to logically establish multiple token-rings in a shared network, which is a multiple group system. We will apply a multiple group method to an $N \times M$ event-builder.

Recall a definition of a DCP group, which is defined in chapter 3. All collectors and members should belong to a group. DCP avoids a packet loss with a token passing mechanism that arbitrates members to transmit data to the collector. If a collector or members belongs to two or more rings, DCP is not able to control members to transmit data to the collector. Therefore, all collectors and members should select just one belonged group.

Since a member belongs to the only one group, a detector subsystem needs to have M members for realizing the $N \times M$ event-builder. These are named a DCP member-cluster.

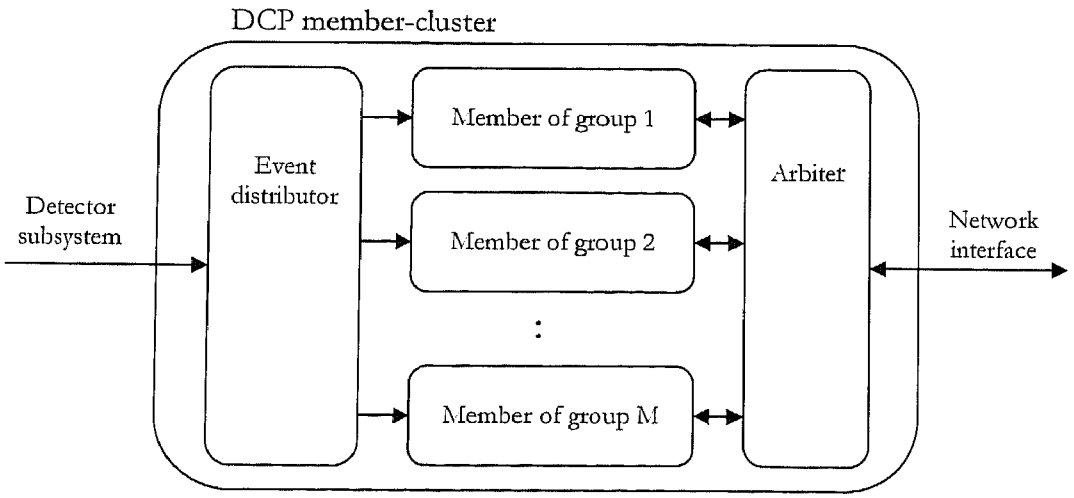

Figure 6.3: DCP member-cluster

Figure 6.3 shows a DCP member-cluster that consists of an event distributor, an arbiter, and M members. The event distributor distributes event-fragment data to members from the detector subsystem with a rule that depends on the system design. The members belongs to a different group each other. Since a cluster has one network interface, the interface is shared by members and two or more members can not use the

interface simultaneously. The arbiter distributes a received and a transmitting packet. Received packets are identified by a group-number in a DCP header and members filter other group's packets with the group-number.

Next, we discuss criteria to achieve high performance in the multiple group system. In the previous section, we discussed the criteria to achieve high performance in a single group system. We extend this idea to a multiple group system. In the multiple group system, two or more groups would use the same line in the multiple group system. Therefore, additional criteria are required.

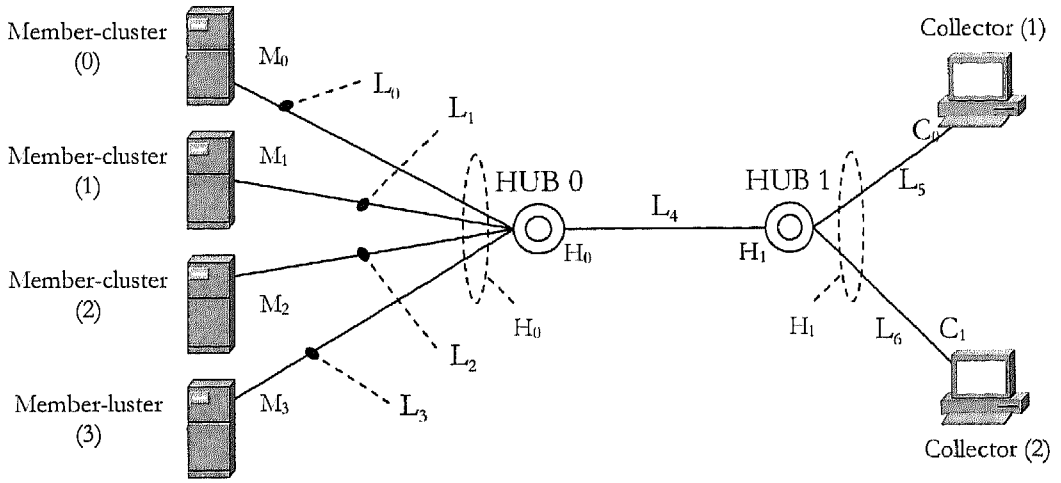

Figure 6.4: 4×2 DCP system

Figure 6.4 shows a 4×2 DCP system and M, H and C denote interface rates and each L represent a line rate. The system consists of four members, two collectors and two HUB's. Collector 1 and 2 belong to group 1 and 2, respectively. All member-clusters have two members which belong to group 1 and 2. We have known the criteria of a single group system, which is

$$L_i \le \min(L_4, L_5, L_6), \quad i = 0,1,2,3.$$

Here, L_i ($0 \le i \le 6$) are line rates as follows:

$$L_i = \min(M_i, H_0), \quad i = 0,1,2,3;$$

$$L_4 = \min(H_0, H_1),$$

$$L_5 = \min(H_1, C_0),$$

$$L_6 = \min(H_1, C_1).$$

The difference from the single group system is that two groups use the same line such as L_4. Since these two groups independently run, L_4 should additionally satisfy the following criteria,

$$L_4 \geq L_j + \max_{i=0,i\neq j}^{3}(L_i)$$

$$L_j \equiv \max_{i=0}^{3}(L_i)$$

Figure 6.5 shows data transfers in this system. The straight a line arrow and a broken arrow display packets of member 1 and 2, respectively.

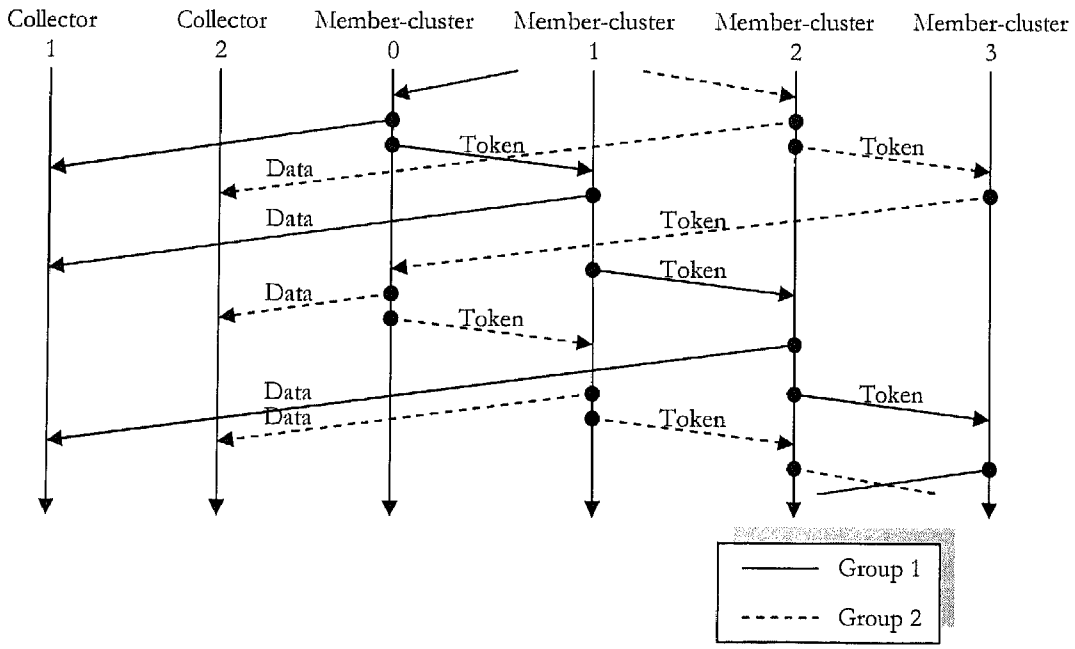

Figure 6.5: Data transfer in the multi group system.

In this system, two sequences of token passing are the same direction and sequence among member-clusters but these are not requirements for the sequences. Therefore, we can define any sequence of a token passing.

6.3 Multiple DCP network system

In a large scale DAQ system, there are many members. In this case, workloads of collectors increase. It is difficult to satisfy criteria to achieve high performance.

A multiple DCP network is most chance to solve the problems. Since the network is hierarchically constructed by multiple DCP networks, the workloads of collectors are distributed and we can easily design networks.

We discuss a multiple DCP network and limit discussion about one group systems for simplicity. The multiple DCP network is a method to hierarchically collect data from many members.

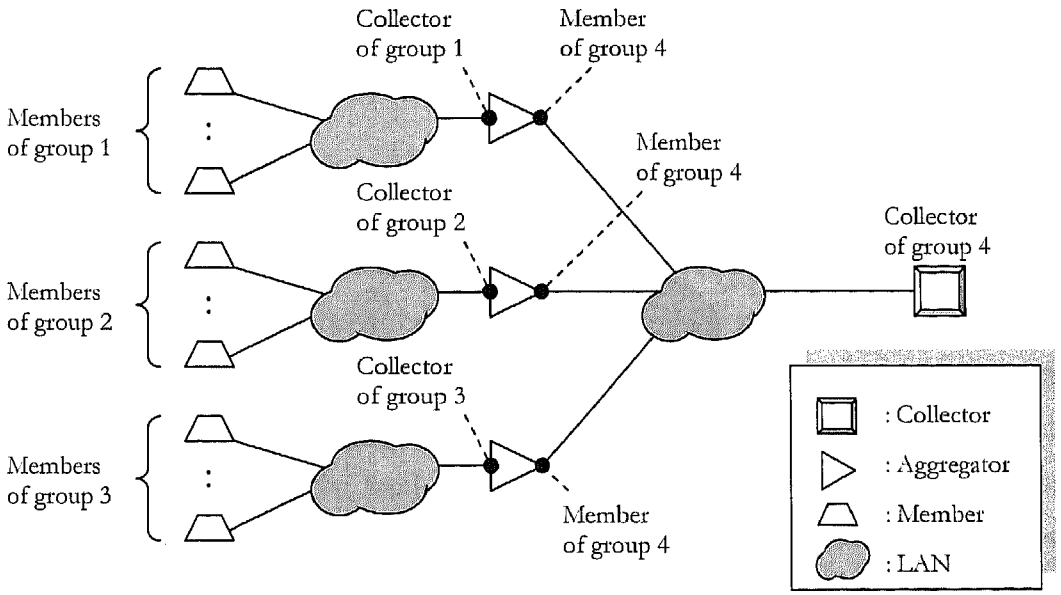

Figure 6.6: Multiple DCP network system

Figure 6.6 shows a multiple DCP network system that consists of four DCP groups. An aggregator has both a member and a collector function, and collects upstream data from members and transmits the data to down stream collector. This system collects many sources from members by two steps. At the first step, source data from members of groups 1 through 3 are collected by collectors of group 1, 2, and 3 in aggregators. At the second step, these aggregated data are collected again in group 4.

We can construct a distributed workload system with this method. It is possible to extend this method to a multiple group system.

6.4 Wide-area DCP network system

In this section, we would discuss problems of a wide-area DCP network system. Recently a size of DAQ systems is becoming large and devices of the system are located in a wide-area. An example of a wide-area DCP network is shown in Figure 6.7. There are many members and collectors. Moreover, the collectors are located far from members. Members and a collector of group i ($0 \leq i \leq 18$) are represented by members i and collector i, respectively.

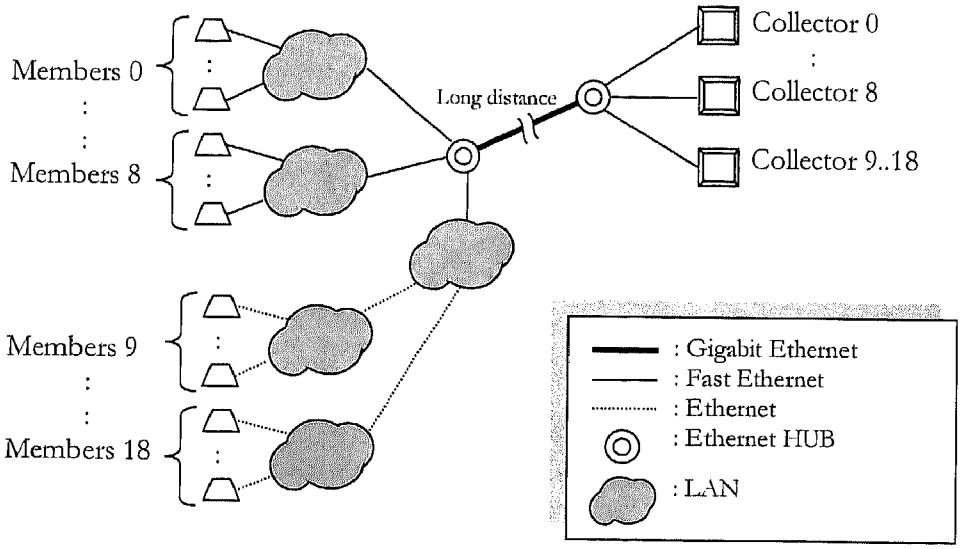

Figure 6.7: A large scale DCP network.

In wide-area DCP network system as this example, it is afraid that system performance decreases for growths of average FTRTT's, the numbers of token-delay points, and delay of data acknowledgement from collectors to members.

We would discuss methods to achieve high performance in the network. First, we discuss a growth of average FTRTT's, the numbers of token-delay points. Recall the following facts.

- The collectors do not pass token packets.
- The average FTRTT is decided by delays of a token passing.
- The number of token-delay points relates with the number of HUB's.

Therefore, the long distances between members and collectors do not impact on performance by the first fact. However, we should design to minimize average

FTRTT's and the number of token-delay points by the second and third facts. Since these two parameters relate with the number of HUB's which is passed by a token packet, we can find that we should design the network to minimize the number of HUB.

Next, we discuss an effect of delay of data acknowledgement. Since DCP adopts a sliding window mechanism for data transfers and the mechanism effectively work under this condition, the delay of data acknowledgement do not seriously impact on the performance.

The example network is designed to achieve high performance. All logical token-ring are in a small LAN for minimizing these FTRTT and the number of token-delay points. The collectors are located far from members but these long distances do not impact on performance. As this example, we can construct a wide-area DCP network and highly efficient data transfer by DCP without complicated traffic engineering as IP networks.

6.5 Application to a HEP experiment

In this section, we discuss an application of DCP to a HEP experiment. The accuracy of HEP measurements becomes more and more precise. In order to improve accuracy of measurements, devices which have fine granularity are developed and many systems have many channels of detectors. In these systems, since a ratio of the number of effective channels and the total number of channels decreases, we need effectively collect data from many sources more than old systems. DCP has a suitable function to effectively collect data from many channels and we adopt DCP for these system.

However, DCP provides an only data transfer function and does not directly process signals from detectors. Therefore, we need an equipment to integrate DCP with front-end devices.

In order to integrate with front-end devices, an electronics readout platform for HEP and nuclear physics experiments is developed at KEK, which is called a COPPER [1] [2]. We apply DCP to a COPPER system. COPPER is a modularized platform and the size of a module is the same size of a single VME 9U board. The module consists of sub-modules and realizes various functions by the platform.

Figure 6.8 and Figure 6.9 show a simplified block diagram and photograph of a COPPER module, respectively. There are seven sub-module slots, which are four A/D

card slots and three PMC card slots which are used for a processor, a trigger, and an application module. The sub-modules are connected via a PCI bus or an original bus, there are two slot types for sub-modules, one is a PCI Mezzanine Card (PMC) [3] type and the other one is an original type. An original-type sub-module is used for an A/D card that is a front-end device such as an Analog-to-Digital-Converter (ADC) and digitizes signals from detectors.

We implement a DCP member on a PMC card as one of the application modules, which is called a DCP network interface card whose photograph is shown in Figure 6.10. The card enables a CPU-less system or reduces a CPU workload. Figure 6.11 shows a block diagram of the DCP network interface card that consists of an FPGA, a FIFO memory, an Ethernet controller, a transformer, and a RJ45 connector.

The DCP network interface card is viewed as a FIFO card from other cards or devices which are connected via the PCI bus. When a device transfers data via the network interface card, the device writes data to the card as a FIFO memory. By this way, the device can transfer data via a DCP network without a processing of network protocols. Therefore, the processing workload of network protocols is reduced and the devices can assign their resources to other tasks, for example, data reduction, data compression, and so on.

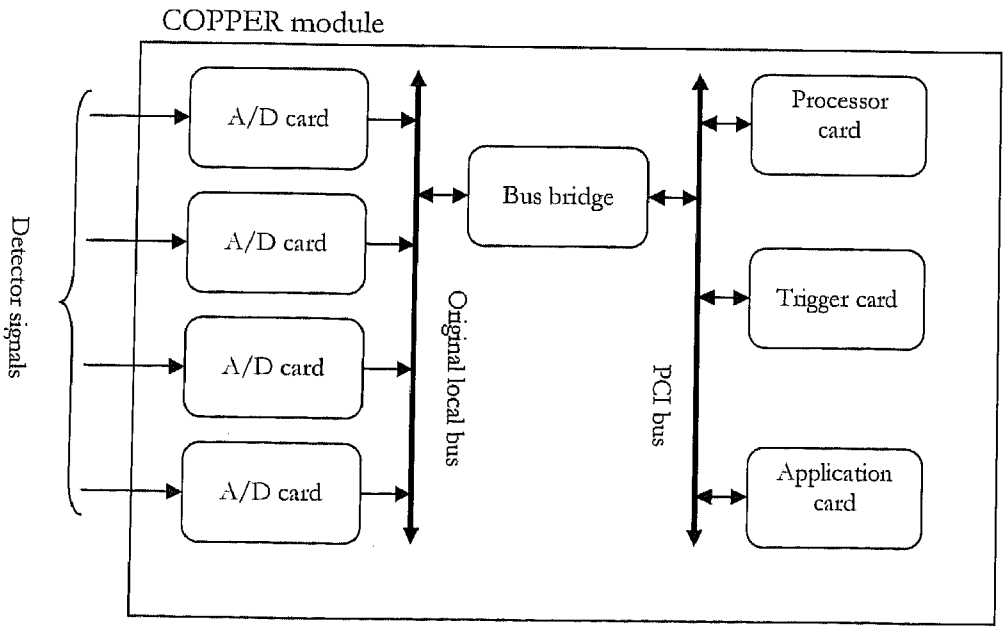

Figure 6.8: A simplified block diagram of a COPPER module.

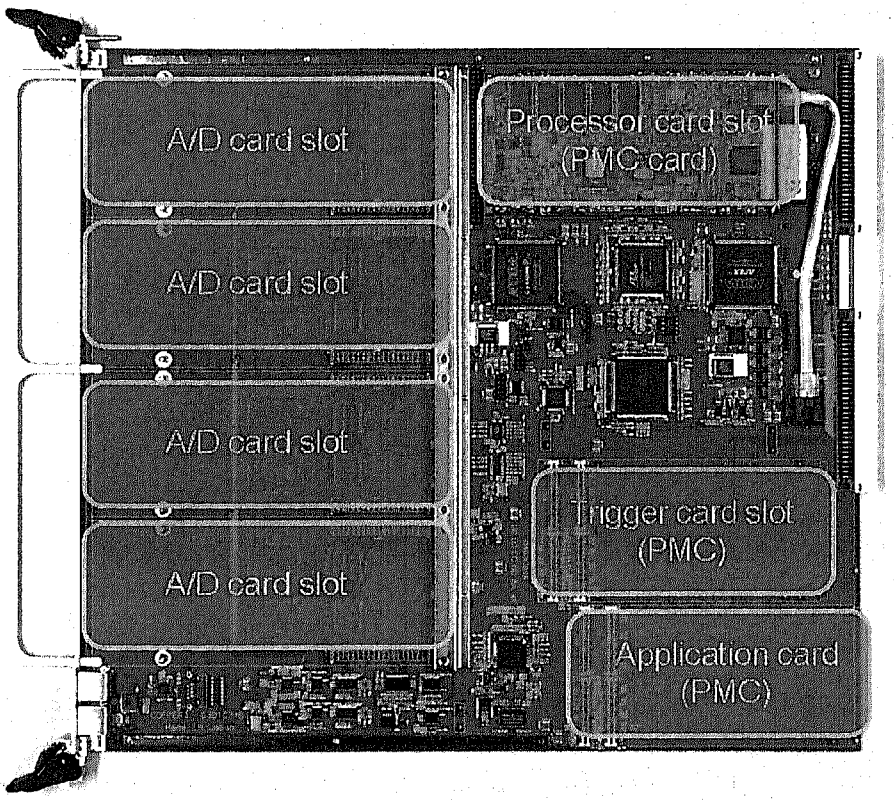

Figure 6.9: Photograph of a COPPER module

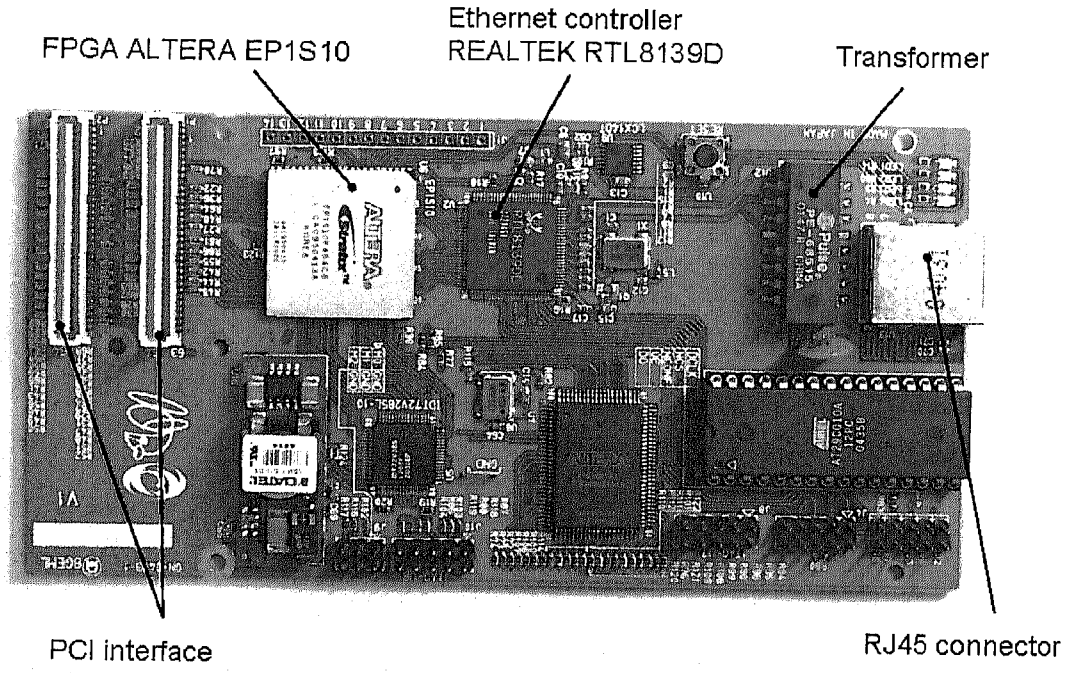

Figure 6.10: Photograph of a DCP network interface card

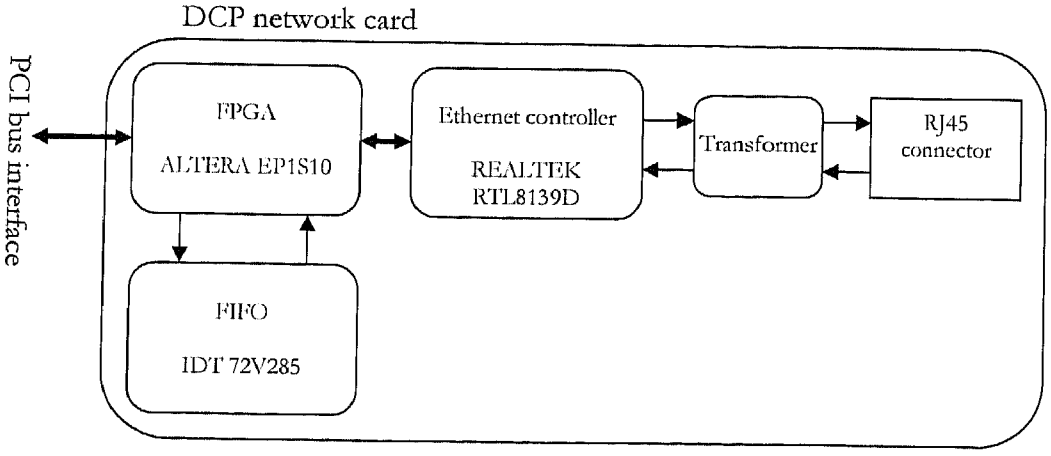

Figure 6.11: Block diagram of a DCP network interface card

References

[1] Y. Igarashi *et al.*, "The Data Acquisition System Based on PMC Bus", Proceedings of the CHEP'03 Conference, eConf C0303241, TUGP009, 2003.

[2] T. Higuchi *et al.*, "Development of a PCI Based Data Acquisition Platform for High Intensity Accelerator Experiments", Proceedings of the CHEP'03 Conference, eConf C0303241, TUGP004, 2003.

[3] IEEE Computer Society P1386.1, "Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC"

Chapter 7

Summary

A network-based DAQ system has been used in high energy physics experiments. Many systems adopt a standard network and a standard reliable protocol.

We pointed out that a DAQ system had unique data flows: in the DAQ system, since an event processor collects event-fragment data from detector subsystems, the detector subsystems could transmit event-fragment data to the event processor at the same time. This situation is not considered in a general network because it rarely occurs in the networks. Therefore, the data flow induces packet losses with a standard reliable protocol in the network. We pointed out that the packet loss made difficulties with a quantitative network design. A standard reliable protocol detects a packet loss at the receiving side and tries to recover the lost data with re-transmissions. Since the protocol efforts to recover errors which are caused of the data transfer under various network conditions, the protocol needs complicated processes, and, then, the behavior becomes entangled. Therefore the packet loss and its behavior in packet switching devices under aggregating many sources are too complicated to predict and moreover its recovery mechanism makes the behavior prediction to be difficult, which makes a performance prediction of data transfer to be unreliable. We can not quantitatively design a network of a DAQ system with a standard reliable protocol, which is a serious problem because it makes to be difficult to design a DAQ system in satisfying requirements of the experiment.

We have developed a new communication network protocol (DCP) for a data acquisition system of HEP experiments, which enables to quantitatively design a network of a DAQ system. The main idea of DCP is to avoid a packet loss with a simple mechanism that is a token passing mechanism because the difficulties with a quantitative network design come from packet losses in a network. By the simple

mechanism, we can easily analyze a performance of DCP with the polling model in queuing theory. Since the polling model is mathematically simplified model, we applied the model to a DCP system and modified equations of the polling model.

DCP has also reliable data delivery. For a reliable data delivery, we introduced a sliding window mechanism that enables to efficiently transfer data. That realizes reliable data delivery in high efficiency. Since our introduced mechanisms are very simple, a total workload of each station is light. We demonstrated the light workload with implementation on a small hardware device.

DCP has the following features.

- Independence from the physical and network layers
- Reliable data delivery
- Scalability
- Quantitative network design
- Light workload

As the above features, DCP has not only reliable delivery as standard protocols but also it enable to quantitatively design the network and reduce workload.

To experimentally verify our model, we measured various DCP systems which are a single HUB system and multiple HUB systems. The single HUB system is the system that has one HUB. The multiple HUB system is the system that has two or more HUB's. These measurement results were in good agreement with expected results by the modified polling model. We demonstrated that the DCP network could be quantitatively designed with the model. We also measured scalability of a DCP system. These measurement results were in good agreement with expected results by the modified polling model. We demonstrated that a DCP system has not a problem about system scalability.

We compared DCP and TCP in terms of performance. We measured transfer rate variations under the situation that a receiver collected data from three senders. The transfer variations with DCP are very smaller than those of TCP. Therefore, we concluded that DCP was suitable protocol for the DAQ system.

Since recently the size of DAQ systems are a growth, we discussed also extension of DCP systems. There are some techniques for the extensions and we discussed it with examples. A multiple group system is useful for construction an N×M

DAQ system. A multiple DCP network system enables to hierarchically construct and overcome difficulties with satisfaction of criteria to achieve high performance. Since recently a size of DAQ systems is becoming large and devices of the system are located in a wide-area, we displayed an example of a wide-area DCP network and pointed out notes in designing the network to achieve high performance. Finally, we demonstrated an application for high energy physics experiments. In the application, DCP enables to release a CPU form heavy workload of protocol processes.

From what has been discussed above, we can conclude that we are able to quantitatively design a high performance DAQ system with DCP.

Appendix A

Ethernet

In this appendix, we review a brief overview of Ethernet. Ethernet is one of Local Area Network (LAN) technologies and is standardized by the Institute of Electrical and Electronics Engineers (IEEE) 802.3 standards committee [1]. At the present, Ethernet is de facto standard interface for PC's, computers and telephones of the IP telephony.

A1 Classical Ethernet

Ethernet was invented by R. Metcalfe and D. Boggs [2] in the early 1970s. Ethernet have a bus topology, which is shown in Figure A.1. There are two configuration types.

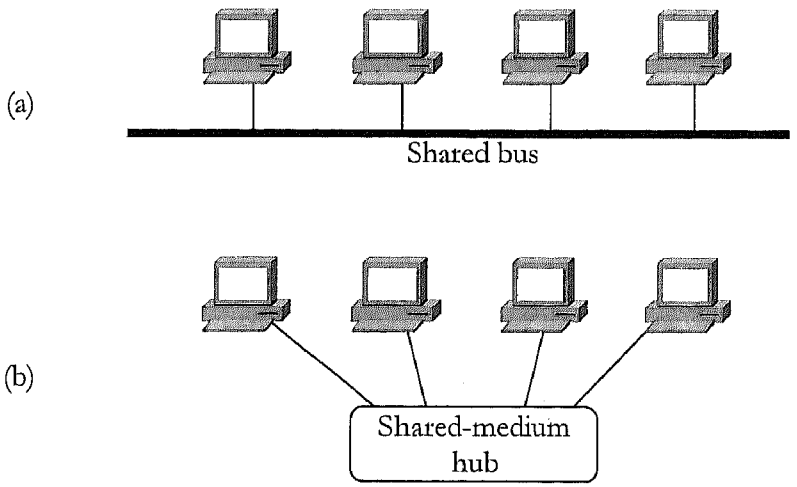

Figure A.1: Bus based LAN

Figure A.1 (a) shows a shared bus configuration. All computers connect to a shared coaxial cable through a Network Interface Card (NIC). Figure A.1 (b) shows a shared hub (HUB) configuration. All computers connect to a shared HUB. The HUB copies a received packet, and, then, transmits the packet to other ports.

Packets which are transmitted by one NIC can be received by all others. This is called broad-cast communication. Since packets contain the address of the destination, only the destination NIC will copy the packet to the main memory of the computer. Since there is no central coordinator to decide which computer can use the shared medium, a distributed approach, called Carrier Sense with Multiple Access/Collision Detection (CSMA/CD), is used. If a NIC wants to transmit a packet, the NIC listens to the medium to check if there is a transmission in progress – this is called carrier sensing. If a transmission is in progress, the NIC waits before attempting again.

It is possible, however, for two NIC's which are far apart in the cable to attempt a transmission at about the same time, detect a free medium, transmit their packets and interfere with each other. This interference is called a collision. Ethernet NIC's are equipped to detect collisions and stop transmitting a packet when collisions are detected. NIC's wait for a randomly selected time period before attempting to re-transmit their packets.

A.2 Frame format

Figure A.1 shows the basic Ethernet frame format. Every frame's transmission is preceded by 64 bits preamble and start frame delimiter during which all other receivers on the media synchronize with the sender. The destination address leads the frame and is read by other Ethernet interfaces in real time to determine whether they are the frame's intended recipient. The source address carries the identity of the frame's source Ethernet interface, and the 64 bit Type field identifies the upper layer protocol whose data make up the payload. After the payload, the frame terminates with a 32 bit Frame Check Sequence. This Ethernet frame is called a MAC (Media Access Control) frame.

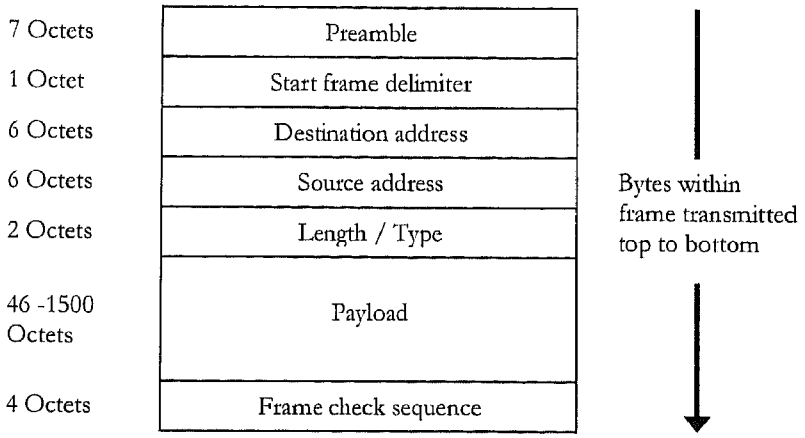

Figure A.1: Basic Ethernet frame format.

A.3 Switching HUB

Since Ethernet use a shared medium, there is always one sender. And in the mechanism, as traffic on an Ethernet network increases, the probability of a collision increases and the network throughput decreases as more of the bandwidth is spent on collisions and retransmissions.

In order to solve this problem, a switching HUB is developed. An incoming frame from a particular station is switched to the appropriate output line to be delivered to the intended destination. At the same time, other unused lines can be used for switching other traffic. The switching HUB has several attractive features:

- No change is required to the software or hardware of the attached devices to convert a bus LAN or a hub LAN to a switched LAN. In the case of an Ethernet LAN, each attached device continues to use the Ethernet medium access control protocol to access the LAN. From the point of view of attached devices, nothing has changed in the access logic.
- Each attached devices has dedicated capacity equal to that of the entire original LAN, assuming that the switching HUB has sufficient capacity to keep up with all attached devices.

- The switching HUB scales easily. Additional devices can be attached to the switching HUB by increasing the capacity of the switching HUB correspondingly.

There are two types of switching HUB's.

- **Store and forward switch:** The switch accepts a frame on an input line, buffers the frame briefly, and then routes the packet to the appropriate output line.
- **Cut through switch:** The switch takes advantage of the fact that the destination address appears at the beginning of the Ethernet frame. The switch begins repeating the incoming frame onto the appropriate output line as soon as the switch recognizes the destination address.

The cut-through switch yields the highest possible throughput but at some risk of propagating bad frames, because the switch is not able to check the FCS before the transmission. The store and forward switch involves a delay between sender and receiver but boosts the overall integrity of the network.

A.4 Current Ethernet

Ethernet has high performance and the interface rate is specified up to 10 Gbit/s, as of Dec. 2004. And also various rates are specified; 10 Mbit/s, 100 Mbit/s, 1 Gbit/s, and 10 Gbit/s. The interface specification is very simple and various types are specified, for example, media types are cuppers and optical fibers. These different interfaces are easily connected via a HUB. Therefore it is possible that there are different interfaces and communicates each other in the same network. Ethernet has sufficiency scalability. There is a limit of the number of connected stations with HUB's but Ethernet has various sub specifications for extensions. Since the mechanism of Ethernet is very simple, the management is also simple. As stated above, since Ethernet has simple functions, the network has been widely adopted various area. The growth of Internet occur widespread use LAN technology, many PC's have an Ethernet interface. There are very large numbers of firms shipping a great variety of products. Large volumes and intense competition have combined to force price down. Consequently, Ethernet devices are very cost effective.

References

[1] IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE standard 802.3, 2003.

[2] R. Metcalfe and D. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks", Comm. ACM, 395-404, 1976.

Appendix B

Polling model

We analyzed a DCP system with a polling model [1] [2] in queuing theory [3]. In this appendix, we derive exact solutions and summarize results of the polling model with gated service.

In the first section, we define the polling model to consider in this appendix. In the second section, we discuss queue lengths at the server arrival to a node. The moments of the queue lengths and an average message length served are derived. In the third section, we derive a queue lengths at service completion of a message is derived. In the fourth section, we derive an average message waiting-time with the average queue length at service completion of a message. In the last section, we summarize notations in this appendix.

B.1 The model

In this section, we introduce the polling model. A basic polling model is a system of multiple queues accessed by a single server in cyclic order. There are three types of a service for each node: an exhaustive service, a gated service, and a limited service. In an exhaustive service system, a queue is served until the queue empties and then a vacation period begins only when there are no messages in the system. Here, we call a time interval when the server is either unavailable or idle a vacation period. In a gated service system, only those messages which are found at the end of a vacation are served and a vacation period begins when the server finish to serve those messages. The messages which arrive to the system during the servicing are reserved for service in the next time. In the limited service system, a server is served until either the queue empties, or the first fixed number of messages is served. In this appendix, we analyze a polling

model with gated service, which is used to analysis a DCP system in this thesis. The following argument is based on the work of O. Hashida [4].

The polling model under consideration is shown in Figure B.1. There N nodes and a server. These nodes served by the server in cyclic order with gated service. The queue capacity at each node is infinite. The arrival process of messages at each node is independent and Poisson random process with arrival rate λ_i.

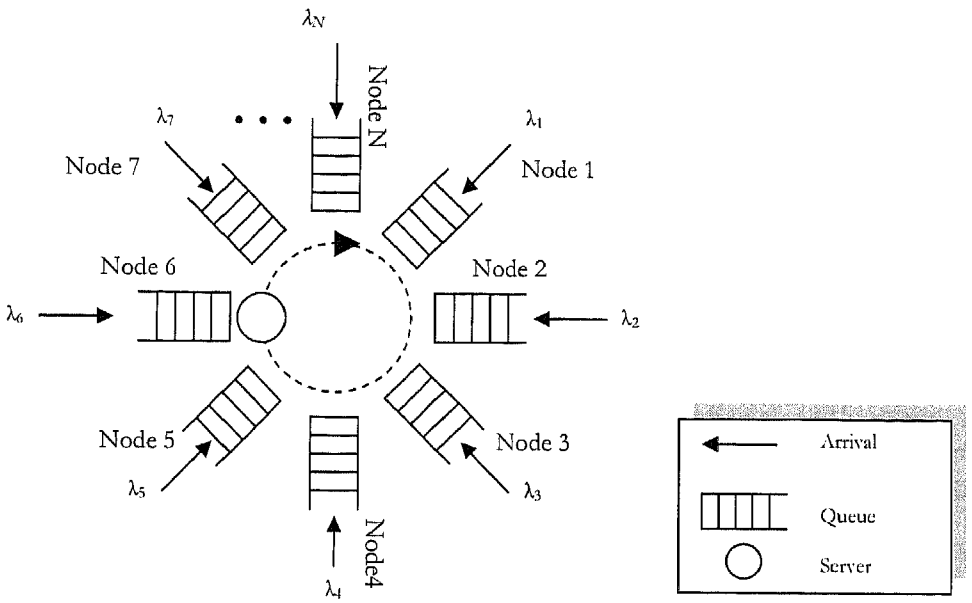

Figure B.1: The polling model.

In this model, we assume that all probability distribution functions have stationary probability distribution functions [5]. Therefore, we can define a generating function [5] of these random variables. The message service time of node i has a probability density distribution function, P_{s_i} and has generating function S_i^*. The random variable of the message service time of node i is denoted by s_i. All message service times are independent and satisfied the following conditions:

$$\overline{s_i} = \int_0^\infty x dP_{s_i}(x) < \infty$$
$$\overline{s_i^2} = \int_0^\infty x^2 dP_{s_i}(x) < \infty \qquad \text{(B.1)}$$

Here, $\bar{s}_i$ and $\overline{s_i^2}$ are the first moment and the second moment of the message service time of node i, respectively. We note that a first moment of a random variable is equal to the average value of the variable. The time to move the server from the node i to node i+1 is called switchover time and has a probability density distribution function, P_{u_i} and has generating function U_i^*. This random variable is denoted by u_i. All switchover times are independent and satisfied the following conditions:

$$\begin{aligned} \bar{u}_i &= \int_0^\infty x dP_{u_i}(x) < \infty \\ \overline{u_i^2} &= \int_0^\infty x^2 dP_{u_i}(x) < \infty \end{aligned} \tag{B.2}$$

Here, $\bar{u}_i$ and $\overline{u_i^2}$ are the first moment and the second moment of the switchover time between node i and node i+1, respectively.

B.2 Queue lengths at the server arrival to a node

In this section, we derive queue lengths at the server arrival to a node. In the first subsection, we derive a generating function of the queue length at the server arrival to a node for deriving the first and second moments of the queue length. In the second subsection we derive the first moments of the queue length with the generating function. In the last subsection, we derive the second moments of the queue length for deriving an average message waiting-time.

B.2.1 Generating function

In this section, we derive a generating function of queue lengths at server arrival to node i, $G^*_i(x_1,x_2, \ldots ,x_N)$, but we can not explicitly derive the generating function. Therefore, we derive the relationship G^*_i and G^*_{i+1}, and, then, we calculate moments of the queue lengths in the next subsection with this relationship. The generating function is defined as follows:

$$\begin{aligned} G^*_i(x_1,x_2,\cdots,x_N) &\equiv \sum_{j_1=0}^{\infty}\cdots\sum_{j_N=0}^{\infty} g_i(j_1,j_1,\cdots j_N)\prod_{l=1}^{N} x_l^{j_l} \\ &= \lim_{n\to\infty}\mathbf{E}\left[\prod_{l=1}^{N} x_l^{\theta_i^n(l)}\right] \end{aligned} \tag{B.3}$$

Here, $g_l(j_1, j_2, \cdots j_N)$ is the probability of queue lengths of node l, $(1 \le l \le N)$, j_l, at the server arrival to node i. $\mathbf{E}(X)$ displays the expectation value of X, which is called the expectation fanction.

We consider an expectation value in the above equation for deriving the relationship G^*_i and G^*_{i+1}. On the other hand, a queue length of node j at the server k-th arrival to node i, $\theta^n_i(k)$, relates $\theta^n_{i+1}(k)$ as the following equations.

$$\theta^n_{i+1}(j) = \theta^n_i(j) + v_j(b_i + u_i), \quad i \neq j;$$
$$\theta^n_{i+1}(j) = v_j(b_i + u_i), \qquad i = j. \tag{B.4}$$

We can rewrite the expectation value with the above equations as follows:

$$\begin{aligned}\mathbf{E}\left[\prod_{k=1}^{N} x_k^{\theta^n_{i+1}(k)}\right] &= \mathbf{E}\left[x_i^{v_i(b_i+u_i)} \prod_{k\neq i} x_k^{\theta^n_i(k)+v_k(b_i+u_i)}\right] \\ &= \mathbf{E}\left[x_i^{v_i(b_i)+v_i(u_i)} \prod_{k\neq i} x_k^{\theta^n_i(k)+v_k(b_i)+v_k(u_i)}\right] \\ &= \mathbf{E}\left[\prod_{k=1}^{N} x_k^{v_k(u_i)}\right] \mathbf{E}\left[x_i^{v_i(b_i)} \prod_{k\neq i} x_k^{\theta^n_i(k)+v_k(b_i)}\right]\end{aligned} \tag{B.5}$$

Next, we calculate the first term of the right-hand side of equation (B.5). Since the random variables, q_i, u_i, are independent variables, the term has a multiple form as

$$\mathbf{E}\left[\prod_{k=1}^{N} x_k^{v_k(u_i)}\right] = \prod_{k=1}^{N} \mathbf{E}\left[x_k^{v_k(u_i)}\right]. \tag{B.6}$$

On the other hand, the expectation function is rewritten by the definition of generating functions. Then

$$\begin{aligned}\mathbf{E}\left[x_k^{v_k(u_i)}\right] &= \sum_{m=0}^{\infty} \mathbf{P}[v_k(u_i) = m] x_k^m \\ &= \sum_{m=0}^{\infty} x_k^m \left[\int_0^{\infty} \frac{(\lambda y)^m}{m!} e^{-\lambda y} u_i(y) dy\right] \\ &= \int_0^{\infty} e^{-\lambda x} \left[\sum_{m=0}^{\infty} \frac{(\lambda y x_k)^m}{m!}\right] u_i(y) dy \\ &= \int_0^{\infty} e^{-\lambda y} e^{\lambda x x_k} u_i(y) dy\end{aligned} \tag{B.7}$$

Here, $\mathbf{P}[v_k(u_i) = m]$ is the probability of $v_k(u_i) = m$ and we use the fact that an arrival process of messages is Poisson arrival process. We further calculate equation (B.6) with equation (B.7) and the result is

$$\begin{aligned}
\mathbf{E}\left[\prod_{k=1}^{N} x_k^{\,v_k(u_i)}\right] &= \prod_{k=1}^{N}\int_0^{\infty} e^{-\lambda y} e^{\lambda x x_k} u_i(y)dy \\
&= \prod_{k=1}^{N}\int_0^{\infty} e^{-\lambda(1-x_k)y} u_i(y)dy \\
&= \int_0^{\infty} e^{-y\sum_{k=1}^{N}\lambda(1-x_k)} u_i(y)dy \quad . \qquad \text{(B.8)} \\
&= U^*{}_i\left(\sum_{k=1}^{N}\lambda(1-x_k)\right) \\
&\equiv U^*{}_i(z)
\end{aligned}$$

This is the final calculated result of the first term of the right-hand side of equation (B.5).

Next, we calculate the second term of the right-hand side of equation (B.5).

$$\begin{aligned}
\mathbf{E}\left[x_i^{\,v_i(b_i)}\prod_{k\neq i} x_k^{\,\theta_i^n(k)+v_k(b_i)}\right] &= \mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} E\left[\prod_{k=1}^{N} x_k^{\,v_k(b_i)} \mid \theta_i^n(1),\theta_i^n(2),\cdots,\theta_i^n(N)\right]\right] \\
&= \mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} E\left[\prod_{k=1}^{N} x_k^{\,v_k(b_i)} \mid \theta_i^n(i)\right]\right] \\
&= \mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} S_i^{*(\theta_i^n(i))}\left(\sum_{k=1}^{N}\lambda(1-q_k)\right)\right] \qquad \text{(B.9)} \\
&= \mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} S_i^*(z)^{\theta_i^n(i)}\right]
\end{aligned}$$

Here, $S_i^{*(x)}$ is the x-th convolution $\underbrace{S_i^* \otimes \cdots \otimes S_i^*}_{x\text{-fold}}$. The above equation is the final calculated result of the second term of the right-hand side of equation (B.5).

We can calculate equation (B.5) with the results (B.8) and (B.9).

$$\mathbf{E}\left[\prod_{k=1}^{N} x_k^{\,\theta_{i+1}^n(k)}\right] = U_i^*(z)\,E\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} S_i^*(z)^{\theta_i^n(i)}\right]. \qquad \text{(B.10)}$$

This is the final calculated result of equation (B.5). We consider limiting $n\to\infty$ of the above equation.

$$\begin{aligned}
\lim_{n\to\infty}\mathbf{E}\left[\prod_{k=1}^{N} x_k^{\,\theta_{i+1}^n(k)}\right] &= \lim_{n\to\infty} U_i^*(z)\,\mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} S_i^*(z)^{\theta_i^n(i)}\right] \\
&= U_i^*(z)\lim_{n\to\infty}\mathbf{E}\left[\prod_{k\neq i} x_k^{\,\theta_i^n(k)} S_i^*(z)^{\theta_i^n(i)}\right]
\end{aligned} \quad . \qquad \text{(B.11)}$$

The both side of the above equation are the same forms of the generating function which is defined by equation (B.3). Therefore, we can rewrite the equation with generating function, thus

$$G^*_{i+1}(x_1,x_2,\cdots,x_N)=U^*_i(z)G^*_i(x_1,\cdots,S^*_i(z),\cdots,x_N). \tag{B.12}$$

This is the final result, which is the relationship G^*_i and G^*_{i+1}.

B.2.2 The first moment

In this section, we calculate moments of the queue length with the relationship which is the final result in the previous section. In this subsection, we calculate the first moment of the queue length of node j at a server arrival to node i. From the nature of generating functions, the first moment, $\bar{g}_j(i)$, which is calculated as follows:

$$\bar{g}_j(i)=\lim_{\substack{x_l\to 1\\ l=1,2,\cdots,N}}\frac{\partial G^*_i(x_1,x_2,\cdots,x_N)}{\partial x_j}. \tag{B.13}$$

First, we differentiate equation (B.12):

$$\frac{\partial G^*_{i+1}}{\partial x_j}=\frac{\partial U^*_i}{\partial x_j}G^*_i+U^*_i\frac{\partial G^*_i}{\partial x_j}. \tag{B.14}$$

If $i\neq j$ then the above equation (B.14) is calculated:

$$\begin{aligned}\partial_j G^*_{i+1}&=\lambda_j(\partial_j U^*_i)G^*_i+U^*_i\left[(\partial_j G^*_i)+\frac{\partial S^*_i}{\partial x_j}\frac{\partial G^*_i}{\partial H^*_i}\right]\\&=\lambda_j(\partial_j U^*_i)G^*_i+U^*_i\left[(\partial_j G^*_i)+\lambda_j(\partial_j S^*_i)(\partial_i G^*_i)\right]\end{aligned}. \tag{B.15}$$

Here, ∂_i is differential of x_i, $\partial/\partial x_i$, and we use the result as follows:

$$\frac{\partial U^*_i}{\partial x_j}=\lambda_j(\partial_j U^*_i),\ \frac{\partial G^*_i}{\partial x_j}=\partial_j G^*_i+\lambda_j(\partial_j S^*_i)(\partial_i G^*_i). \tag{B.16}$$

We obtain the relationship of the first moments with equation (B.15) when $i\neq j$:

$$\bar{g}_j(i+1)=\lambda_j\bar{u}_i+\bar{g}_j(i)+\lambda_j\bar{s}_i\bar{g}_i(i). \tag{B.17}$$

Here, we use nature of generating function as follows:

$$\begin{aligned}&U^*_i(1)=1,\ \partial_i U^*_i(1)=\bar{u}_i;\\&G^*_i(1)=1,\ \partial_j G^*_i(1)=\bar{g}_j(i).\end{aligned} \tag{B.18}$$

We sum equation (B.17) over $i=j,j+1,\cdots,i-1$. This yields

$$\bar{g}_j(i) = \lambda_j \sum_{k=j}^{i-1} \left[\bar{u}_k + \bar{s}_k \bar{g}_k(k) \right]. \tag{B.19}$$

If $i = j$ then equation (B.14) is calculated in a similar fashion:

$$\begin{aligned} \partial_j G^*_{j+1} &= \lambda_j \partial_j U^*_j G^*_j + \frac{\partial S^*_j}{\partial x_j} \frac{\partial G^*_j}{\partial H^*_j} \\ &= \lambda_j \partial_j U^*_j G^*_j + \lambda_j \partial_j S^*_j \partial_j G^*_j \end{aligned} . \tag{B.20}$$

We obtain the relationship of the first moments with equation (B.15) when $i = j$:

$$\bar{g}_j(j+1) = \lambda_j \bar{u}_j + \lambda_j \bar{s}_j \bar{g}_j(j). \tag{B.21}$$

We sum the above equation over $i = 1,2,\cdots N$. This yields

$$\bar{g}_j(j) = \lambda_j \sum_{k=1}^{N} \left[\bar{u}_k + \bar{s}_k \bar{g}_k(k) \right]. \tag{B.22}$$

The equation (B.19) and equation (B.22) is summarized with the matrix form as the following:

$$\begin{pmatrix} 1-\lambda_1 \bar{s}_1 & -\lambda_1 \bar{s}_2 & \cdots & -\lambda_1 \bar{s}_N \\ -\lambda_2 \bar{s}_1 & 1-\lambda_2 \bar{s}_2 & \cdots & -\lambda_2 \bar{s}_N \\ \vdots & \vdots & & \vdots \\ -\lambda_N \bar{s}_1 & -\lambda_N \bar{s}_2 & \cdots & 1-\lambda_N \bar{s}_N \end{pmatrix} \begin{pmatrix} \bar{g}_1(1) \\ \bar{g}_2(2) \\ \vdots \\ \bar{g}_N(N) \end{pmatrix} = \sum_{k=1}^{N} u_k^{(1)} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{pmatrix}. \tag{B.23}$$

The solution of the above equations is an average message length of node i, L_i, because a server processes messages in the queue of node i at the server arrival to node i. The solution is

$$\begin{aligned} \bar{g}_i(i) &= \frac{\lambda_i \sum_{k=1}^{N} \bar{u}_k}{1 - \sum_{k=1}^{N} \lambda_k \bar{s}_k} \\ &\equiv L_i \end{aligned} . \tag{B.24}$$

By the above equation (B.24) and equation (B.19), we can find that the first moment of a queue length of node j at a server arrival to node i is

$$\bar{g}_j(i) = \lambda_j \sum_{k=j}^{i-1} \left[\bar{u}_k + \bar{s}_k L_k \right]$$

$$= \lambda_j \sum_{k=j}^{i-1} \left[\bar{u}_k + \frac{\lambda_k \bar{s}_k \sum_{l=1}^{N} \bar{u}_l}{1 - \sum_{l=1}^{N} \lambda_l \bar{s}_l} \right]. \tag{B.25}$$

In this equation, the first term and the second term of right-hand side mean the average number of arrival customers during switchover times and processing times, respectively.

Equations (B.24) and (B.25) are the final results in this subsection. We use the result in the following subsection.

B.2.3 The second moment

From this subsection, we limit discuss to analysis a symmetric polling system for simplicity. In this system, all nodes have the same statistics nature and we assume as the following conditions for $0 \le i \le N$:

$$\begin{aligned} \lambda_i &= \lambda, \\ U_i &= U, \\ S_i &= S. \end{aligned} \tag{B.26}$$

By the above assumption, the following equations are satisfied for all i in this system.

$$\begin{aligned} \bar{u}_i &= \bar{u}, \ \overline{u_i^2} = \overline{u^2}, \\ \bar{s}_i &= \bar{s}, \ \overline{s_i^2} = \overline{s^2}. \end{aligned} \tag{B.27}$$

In this subsection, we calculate the second factorial moment of a queue length of node i at a server arrival to node i, $g_{i,i}^{(2)}(i)$, for calculation of an average message waiting-time in the next section. In the symmetric system, the second factorial moment $g_{i,i}^{(2)}(i)$ is satisfied $g_{i,i}^{(2)}(i) = g^{(2)}$ because all nodes have the same condition.

From the nature of generating functions, the second factorial moment, $g_{j,k}^{(2)}(i)$, is calculated as follows:

$$g_{j,k}^{(2)}(i) = \lim_{\substack{x_l \to 1 \\ l=1,2,\cdots,N}} \frac{\partial^2 G_i^*(x_1, x_2, \cdots, x_N)}{\partial x_j \partial x_k}. \tag{B.28}$$

We consider three differential conditions: the first case is $j \neq k$, $i \neq j,k$; the second case is $i = k$, $i \neq j$; and the third case is $i = j = k$.

In the first case, $j \neq k$, $i \neq j,k$. The result is

$$\begin{aligned} g_{j,k}^{(2)}(i+1) = {} & \lambda^2 \overline{u^2} + \lambda \bar{u}\left[\bar{g}_k(i) + \bar{g}_j(i)\right] \\ & + \lambda^2\left(2\overline{su} + \overline{s^2}\right)\bar{g}_i(i) \\ & + \lambda \bar{s}\left[g_{i,k}^{(2)}(i) + g_{j,i}^{(2)}(i)\right] \\ & + \left(\lambda \bar{s}\right)^2 g^{(2)} + g_{j,k}^{(2)}(i) \end{aligned} \quad . \tag{B.29}$$

In the second case, $i = k$, $i \neq j$. The result is

$$\begin{aligned} g_{j,i}^{(2)}(i+1) = {} & \lambda^2 \overline{u^2} + \lambda \overline{u}\,\bar{g}_j(i) \\ & + \lambda^2\left(2\overline{us} + \overline{s^2}\right)\bar{g}_i(i) \\ & + \lambda \bar{s} g_{j,i}^{(2)}(i) \\ & + \left(\lambda \bar{s}\right)^2 g^{(2)} \end{aligned} \quad . \tag{B.30}$$

In the third case, $i = j = k$. The result is

$$\begin{aligned} g_{j,j}^{(2)}(j+1) = {} & \lambda^2 \overline{u^2} \\ & + \left(2\lambda^2 \overline{us} + \lambda^2 \overline{s^2}\right)\bar{g}_j(j) \\ & + \left(\lambda \bar{s}\right)^2 g^{(2)} \end{aligned} \quad . \tag{B.31}$$

Next, we will calculate $g^{(2)}$. We sum $g_{j,k}^{(2)}(i+1)$ over applicable i and k.

If $j \neq k$, then we sum over $g_{j,k}^{(2)}(i+1)$ applicable i with equations (B.29) and (B.30).

$$\sum_{i=1}^{N} g_{j,k}^{(2)}(i+1) = \sum_{i=1, i\neq j,k}^{N} g_{j,k}^{(2)}(i+1) + g_{j,k}^{(2)}(j+1) + g_{j,k}^{(2)}(k+1)$$

$$\begin{aligned} g_{j,k}^{(2)}(j) + g_{j,k}^{(2)}(k) = {} & N\lambda^2 \overline{u^2} + \lambda \bar{u}\left[\sum_{i=1, i\neq k}^{N} \bar{g}_k(i) + \sum_{i=1, i\neq j}^{N} \bar{g}_j(i)\right] \\ & + \lambda^2\left(2\overline{su} + \overline{s^2}\right)\sum_{i=1}^{N} \bar{g}_i(i) \\ & + \lambda \bar{s}\left[\sum_{i=1, i\neq k}^{N} g_{i,k}^{(2)}(i) + \sum_{i=1, i\neq j}^{N} g_{j,i}^{(2)}(i)\right] \\ & + N\left(\lambda \bar{s}\right)^2 g^{(2)} \end{aligned} \quad . \tag{B.32}$$

The symmetric system satisfies the following condition:

$$g^{(2)} = g_{i,i}{}^{(2)}(i),\ \bar{L} = \bar{g}_i(i),$$

$$\sum_{i=1}^{N} \bar{g}_i(j) = \sum_{j=1}^{N} \bar{g}_j(i) \equiv \Sigma^{(1)}, \tag{B.33}$$

$$\sum_{j=1}^{N} g_{i,j}{}^{(2)}(j) = \sum_{j=1}^{N} g_{j,i}{}^{(2)}(j) \equiv \Sigma^{(2)}(i).$$

We rewrite equation (B.32) with the above notation:

$$\begin{aligned} g_{j,k}{}^{(2)}(j) + g_{j,k}{}^{(2)}(k) = N\lambda^2\overline{u^2} + 2\lambda\bar{u}\Sigma^{(1)} + L\left[\lambda^2 N\left(2\overline{su} + \overline{s^2}\right) - 2\lambda\bar{u}\right] \\ + \lambda\bar{s}\left[\Sigma^{(2)}(k) + \Sigma^{(2)}(j)\right] + g^{(2)}\left[N\left(\lambda\bar{s}\right)^2 - 2\lambda\bar{s}\right] \end{aligned} \tag{B.34}$$

In the case of $j = k$, we calculate a sum of $g_{i,j}{}^{(2)}(i+1)$ in a similar fashion and sum $g_{j,j}{}^{(2)}(i+1)$ over applicable i with equation (B.31).

$$\begin{aligned} \sum_{i=1}^{N} g_{k,k}{}^{(2)}(i+1) &= \sum_{i=1, i\neq k}^{N} g_{k,k}{}^{(2)}(i+1) + g_{k,k}{}^{(2)}(k+1) \\ g_{k,k}{}^{(2)}(k) &= N\lambda^2\overline{u^2} + \lambda\bar{u}\left[\sum_{i=1, i\neq k}^{N} \bar{g}_k(i) + \sum_{i=1, i\neq j}^{N} \bar{g}_j(i)\right] \\ &\quad + \lambda^2\left(2\overline{su} + \overline{s^2}\right)\sum_{i=1}^{N} \bar{g}_i(i) \\ &\quad + \lambda\bar{s}\left[\sum_{i=1, i\neq k}^{N} g_{i,k}{}^{(2)}(i) + \sum_{i=1, i\neq j}^{N} g_{j,i}{}^{(2)}(i)\right] \\ &\quad + N\left(\lambda\bar{s}\right)^2 g^{(2)} \end{aligned} \tag{B.35}$$

Rewriting the above equation with equation (B.33) we have

$$\begin{aligned} g_{k,k}{}^{(2)}(k) = N\lambda^2\overline{u^2} + 2\lambda\bar{u}\Sigma^{(1)} + L\left[\lambda^2\left(2\overline{su} + \overline{s^2}\right) - 2\lambda\bar{u}\right] \\ + 2\lambda\bar{s}\Sigma^{(2)}(k) + g^{(2)}\left[N\left(\lambda\bar{s}\right)^2 - 2\lambda\bar{s}\right] \end{aligned} \tag{B.36}$$

Next, we sum $\overline{g_{j,k}{}^2}(i+1)$ over applicable k with equations (B.34) and (B.36). The result is

$$\begin{aligned} 2\Sigma^{(2)} = N^2\lambda^2\overline{u^2} + 2\lambda\bar{u}\Sigma^{(1)} + NL\left[N\lambda^2\left(2\overline{su} + \overline{s^2}\right) - 2\lambda\bar{u}\right] \\ + 2N\lambda\bar{s}\Sigma^{(2)} + g^{(2)}\left\{N\left[\left(\lambda\bar{s}\right)^2 - 2\lambda\bar{s}\right] + 1\right\} \end{aligned} \tag{B.37}$$

We obtain the final result, $\overline{g^2}$, to substitute equation (B.37) into equation (B.36). We have

$$g^{(2)} = \frac{N\lambda^2\overline{u^2} + 2\lambda\bar{u}\Sigma^{(1)} + L\left|N\left(2\lambda^2\overline{su} + \lambda^2\overline{s^2}\right) - 2\lambda\bar{u}\right|}{\left(1-N\lambda\bar{s}\right)\left(1+\lambda\bar{s}\right)}$$
$$= \frac{N\lambda^2\overline{u^2} + \dfrac{N^2\lambda^3\bar{u}\overline{s^2}}{1-N\lambda\bar{s}} + N\lambda^2\bar{u}^2\left[\dfrac{N\left(1+\lambda\bar{s}\right)}{1-N\lambda\bar{s}} - 1\right]}{\left(1-N\lambda\bar{s}\right)\left(1+\lambda\bar{s}\right)} \quad \text{(B.38)}$$

We note that the second factorial moment has a relation, $g^{(2)} \equiv \overline{g^2} - \bar{g}$, by the definition.

B.3 Queue lengths at service completion of a message

In this section, we derive the first moment of a queue length of node i at service completion of a message of node i. In order to derive that moment, first, we derive a generating function of the queue length. Second, we derive the first moment with the generating function.

B.3.1 Generating function

In this subsection, we derive the generating function of the queue length of node j at service completion of a message. This generating function $Q_i^*(x_1, x_2, \cdots, x_N)$, is defined by

$$Q_i^*(x_1, x_2, \cdots, x_N) \equiv \mathbf{E}\left(\prod_{j=1}^{N} x_j^{\xi_j^{(m)}(i)}\right). \quad \text{(B.39)}$$

Here, $\xi_j^{(m)}(i)$ is a queue length of node j at the time t_m when is the service completion time of m-th message in the queue of node i at the server arrival to node i. We consider the queue length under the system state $(\theta_1(i), \theta_2(i), \cdots, \theta_N(i))$ at the server arrival to node i. Here, $\theta_j(i)$ is the queue length of node j at the server arrival to node i. Under the condition which is discussed above, the time t_m is satisfied the following equation:

$$t_k = t_a + \tau_m. \quad \text{(B.40)}$$

Here, t_a is the arrival time of the server to node i and τ_m is the interval time from the arrival time to the service completion of m-th message. By the above equation, we obtain a relationship equation of the queue length of node j at the time t_m. We have

$$\xi_i^{(m)}(i) = \theta_i(i) - m + \nu_i(\tau_m),\ i = j,\ m \le \theta_i(i);$$
$$\xi_j^{(m)}(i) = \theta_j(i) + \nu_j(\tau_m),\ i \ne j. \tag{B.41}$$

Since we consider the system under the condition $(\theta_1(i), \theta_2(i), \cdots, \theta_N(i))$, we can rewrite the above equation with a conditional expectation:

$$Q_i^*(x_1, x_2, \cdots, x_N) =$$
$$K \sum_{\theta_1(i)=0}^{\infty} \cdots \sum_{\theta_N(i)=0}^{\infty} g_i(\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \times \mathbf{E}\left(\prod_{j=1}^{N} x_j^{\theta_j^{(m)}(i)} \middle| (\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \right). \tag{B.42}$$

Here, K is a constant for probability normalization and $g_i(\theta_1(i), \theta_1(i), \cdots \theta_N(i))$ represents probability of system state $(\theta_1(i), \theta_2(i), \cdots, \theta_N(i))$ at the server arrival to node i. We further calculate the expectation of equation (B.42) with equation (B.41):

$$\begin{aligned}
&\mathbf{E}\left[\prod_{j=1}^{N} x_j^{\theta_j^{(m)}(i)} \middle| (\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \right] \\
&= \mathbf{E}\left[x_i^{\theta_i(i) - m + \nu_i(\tau_m)} \prod_{j=1, j \ne i}^{N} x_j^{\theta_j(i) + \nu_j(\tau_m)} \middle| (\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \right] \\
&= \mathbf{E}\left[x_i^{-m} \prod_{j=1}^{N} x_j^{\theta_j(i) + \nu_j(\tau_m)} \middle| (\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \right] \\
&= x_i^{-m} \prod_{j=1}^{N} x_j^{\theta_j(i)} \mathbf{E}\left[\prod_{k=1}^{N} x_k^{\nu_k(\tau_m)} \middle| m \right] \\
&= x_i^{-m} \prod_{j=1}^{N} x_j^{\theta_j(i)} S_i^{*(m)}\left(\sum_{k=1}^{N} \lambda(1 - x_k) \right) \\
&= x_i^{-m} \prod_{j=1}^{N} x_j^{\theta_j(i)} \left[S_i^*(z) \right]^m
\end{aligned} \tag{B.43}$$

Here, we use independency of $\theta_i(i)$ and $\nu_i(\tau_k)$, the definition of the generating function of a service time, and define $z \equiv \sum_{k=1}^{N} \lambda(1 - x_k)$. We substitute the above equation (B.43) into equation (B.42), thus

$$Q_i^*(x_1, x_2, \cdots, x_N) = K \sum_{\theta_1(i)=0}^{\infty} \cdots \sum_{\theta_N(i)=0}^{\infty} g_i(\theta_1(i), \theta_2(i), \cdots, \theta_N(i)) \times \prod_{j=1}^{N} x_j^{\theta_j(i)} \times \sum_{k=1}^{\theta_i(i)} \left[\frac{S_i^*(z)}{x_i} \right]^k. \tag{B.44}$$

We calculate the last summation of the right-hand side of the above equation:

$$
\begin{aligned}
&Q_i^*(x_1,x_2,\cdots,x_N)\\
&=K\sum_{\theta_1(i)=0}^{\infty}\cdots\sum_{\theta_N(i)=0}^{\infty}g_i(\theta_1(i),\theta_2(i),\cdots,\theta_N(i))\times\prod_{j=1}^{N}x_j^{\theta_j(i)}\left\{1-\left[\frac{S_i^*(z)}{x_i}\right]^{\theta_i(i)}\right\}\frac{S_i^*(z)}{x_i-S_i^*(z)}\\
&=K\sum_{\theta_1(i)=0}^{\infty}\cdots\sum_{\theta_N(i)=0}^{\infty}g_i(\theta_1(i),\theta_2(i),\cdots,\theta_N(i))\times\left\{\prod_{j=1}^{N}x_j^{\theta_j(i)}-\prod_{l=1,l\neq i}^{N}x_l^{\theta_l(i)}\times S_i^*(z)^{\theta_i(i)}\right\}\frac{S_i^*(z)}{x_i-S_i^*(z)}
\end{aligned}
\tag{B.45}
$$

Recall the definition of a generating function which is equation (B.3). We can further calculate the above equation with the definition of G_i^*:

$$
Q_i^*(x_1,x_2,\cdots,x_N)=K\frac{S_i^*(z)}{x_i-S_i^*(z)}\left\{G_i^*(x_1,x_2,\cdots,x_N)-G_i^*\left(x_1,x_2,\cdots,S_i^*(z),\cdots,x_N\right)\right\}. \tag{B.46}
$$

The above equation is the final result in this subsection. We will calculate first moment of the queue length of node j at service completion of a message in the next subsection with this result. The coefficient K will be calculated later.

B.3.2 The first moment

We calculate the first moment of the queue length of node i at service completion of a message in the queue of node i before calculating the constant K.

$$
\begin{aligned}
\overline{q_i}&=\lim_{x_j\to 1,j=1,2,\cdots N}\frac{\partial Q_i^*}{\partial x_i}\\
&=\lim_{x_j\to 1,j=1,2,\cdots N}K\frac{(x_i-S_i^*(z))\left[G_i^*(x_1,x_2,\cdots,x_N)\right]'-(x_i-S_i^*(z))'\left[G_i^*(x_1,x_2,\cdots,S_i^*(z),\cdots,x_N)\right]}{\left[x_i-S_i^*(z)\right]^2}
\end{aligned}
\tag{B.47}
$$

To further calculate the above equation, we use L'Hospital's theorem, thus

$$
\begin{aligned}
\overline{q_i}&=\lim_{x_j\to 1,j=1,2,\cdots N}K\frac{\frac{\partial^2}{\partial x_i^{\,2}}\left\{(x_i-S_i^*(z))\left[G_i^*(x_1,x_2,\cdots,x_N)\right]'-(x_i-S_i^*(z))'\left[G_i^*(x_1,x_2,\cdots,S_i^*(z),\cdots,x_N)\right]\right\}}{\frac{\partial^2}{\partial x_i^{\,2}}\left\{\left[x_i-S_i^*(z)\right]^2\right\}}\\
&=K\frac{(1-\lambda\bar{s})\left\{2\lambda\bar{s}\,\overline{g_i}(i)(1-\lambda\bar{s})+g_{i,i}^{(2)}(i)\left[1-(\lambda\bar{s})^2\right]\right\}}{2(1-\lambda\bar{s})^2}\\
&=K\left[\lambda\bar{s}\,\overline{g_i}(i)+\frac{(1+\lambda\bar{s})g_{i,i}^{(2)}(i)}{2}\right]
\end{aligned}
\tag{B.48}
$$

This is the first moment of the queue length of node i at service completion of a message in the queue of node i.

We may evaluate the coefficient K in the above equation by recognizing that $Q_i^*(1,\cdots,1)=1$ and using L'Hospital's theorem, thus

$$\begin{aligned}
&\lim_{x_j\to 1, j=1,2,\cdots N} Q(x_1,x_2,\cdots,x_N) \\
&= \lim_{x_j\to 1, j=1,2,\cdots N} K\frac{\frac{\partial}{\partial x_i}\left\langle S_i^*(z)\left\{G_i^*(x_1,x_2,\cdots,x_N)-G_i^*\left(x_1,x_2,\cdots,S_i^*(z),\cdots,x_N\right)\right\}\right\rangle}{\frac{\partial}{\partial x_i}\left[x_i-S_i^*(z)\right]} \qquad \text{(B.49)} \\
&= K\bar{g}_i(i) \\
&= 1
\end{aligned}$$

$$\Rightarrow K=\frac{1}{\bar{g}_i(i)}.$$

We obtain the first moment of the queue length of node i at service completion of a message of node i as the following equation:

$$\bar{q}_i=\lambda\bar{s}+\frac{\left(1+\lambda\bar{s}\right)g_{i,i}^{(2)}(i)}{2\bar{g}_i(i)}. \qquad \text{(B.50)}$$

In the above equation, the first term of the right-hand side means an average number of messages in the server and the second term means an average number of messages in the queue. Since we consider the symmetric system, we rewrite the above equation with equation (B.24) and equation (B.38). We obtain final expression of the first moment in the symmetric system, Q, which is

$$\begin{aligned}
Q &= \lambda\bar{s}+\frac{\lambda\left(\overline{u^2}-\bar{u}^2\right)}{2\bar{u}}+\frac{\lambda\left[N\lambda\overline{s^2}+N\bar{u}\left(1+\lambda\bar{s}\right)\right]}{2\left(1-N\lambda\bar{s}\right)} \\
&= \lambda\bar{s}+\frac{\lambda\sigma_u^{\,2}}{2\bar{u}}+\frac{\lambda\left[N\lambda\overline{s^2}+N\bar{u}\left(1+\lambda\bar{s}\right)\right]}{2\left(1-N\lambda\bar{s}\right)}
\end{aligned} \qquad \text{(B.50)}$$

Here, $\sigma_u^{\,2}$ is variance of u.

B.4 Message waiting-time

We derive an average message waiting-time. In the previous section, we derived the first moment of the queue length of node i at service completion of a

message of node i. The average message waiting-time in the queue of node i relate the first moment of the queue length through a Little's Law [3] which is represented by

$$Q = \lambda W . \tag{B.51}$$

This law states that the average number of customers in a queuing system, Q, is equal to the average arrival rate of customers to that system, λ, times the average time spent in that system, W.

We drive the average message waiting-time by using the Little's Law, thus

$$W = \bar{s} + \frac{\sigma_u^2}{2\bar{u}} + \frac{\left|N\lambda\overline{s^2} + N\bar{u}\left(1+\lambda\bar{s}\right)\right|}{2\left(1-N\lambda\bar{s}\right)}. \tag{B.52}$$

This is the final result. Rewriting the above equation with notation of this thesis:

$$\begin{aligned}\overline{W} &= \bar{s} + \frac{\sigma_u^2}{2\bar{u}} + \frac{\left|N\lambda\overline{s^2} + N\bar{u}\left(1+\lambda\bar{s}\right)\right|}{2(1-\rho)} \\ &= \bar{s} + \frac{\sigma_u^2}{2\bar{u}} + \frac{1}{2(1-\rho)}\left[N\bar{u}\left(1+\lambda\bar{s}\right) + N\lambda\bar{s}\frac{\overline{s^2}}{\bar{s}}\right] \\ &= \bar{s} + \frac{\sigma_u^2}{2\bar{u}} + \frac{1}{2(1-\rho)}\left[N\bar{u}\left(1+\lambda\bar{s}\right) + \rho\frac{\overline{s^2}}{\bar{s}}\right]\end{aligned} \tag{B.53}$$

Here, ρ is total server utilization which is defined by

$$\rho \equiv N\lambda\bar{s} . \tag{B.54}$$

B.5 Notation

We summarize notations in this appendix as follows:

$\bar{y}$:	First moment of the random variable y. This value is equal to the average of the random variable y.
$\overline{y^2}$:	Second moment of the random variable y.
$G^{(2)}$:	Second factorial moment of a generating function G.
s_i:	Service time of node i, which is a random variable.
s:	Service time of node i in a symmetric system, which is a random variable.

u_i:	Switchover time between node i-1 and node i, which is a random variable.
u:	Switchover time between nodes in a symmetric system, which is a random variable.
$v_i(t)$:	Number of arrived messages to node i during t, which is a random variable.
x_i:	Variable of node i of a generating function.
λ_i:	Arrival rate of node i with Poisson random process.
λ:	Arrival rate of a node with Poisson random process in a symmetric system.
$g_i(j_1, j_2, \cdots, j_N)$:	Probability of queue lengths of node l $(1 \le l \le N)$, j_l, at the server arrival to node i in equilibrium, which is a stationary probability distribution function.
$G_i^*(x_1, x_2, \cdots, x_N)$:	Generating function of queue lengths of nodes at the server arrival to node i.
L_j:	Average message length of node j.
L:	Average message length of a node in a symmetric system.
$P_y(\bullet)$:	Probability density distribution for the random variable y.
$q_j(i)$:	Queue length of node j at service completion of a message in the queue of node i, which is a random variable.
$Q_i^*(j_1, j_2, \cdots j_N)$:	Generating function of queue lengths of nodes at service completion of a message in the queue of node i.
W:	Average message waiting-time of a node in a symmetric system.

References

[1] H. Takagi, "Queuing analysis of polling models", ACM Computing Surveys, 20, 5-28, 1988.

[2] H. Takagi, "Analysis and application of polling models", Institute of Policy and Planning Sciences discussion paper series No.805, University of Tsukuba, 1998.

[3] L. Kleinrock, "Queuing system, Volume 1: Theory", Jhon Wiley and Sons, 1975.

[4] O. Hashida, "Gating multiqueues served in cyclic order", Trans. Inst. Electron. Commun. Eng. Jpn. 53-A, 43-50, 1970.

[5] G. Grimmett and D. Stirzaker, "Probability and Random Processes, Third Edition", Oxford, 2001.