VLSI Design and Implementation of Multimedia Transport Protocol for Reliable Networks

장 종 욱*

Jong-Wook Jang*

요 약

앞으로의 트랜스포트 프로토콜은 Gbps이상의 처리율, 다양한 기능 및 응용에 따른 융통성 있는 적용이 가능해야 한다. 이를 위해서 새로운 트랜스포트 프로토콜의 개발 또는 VLSI를 이용한 프로토콜의 하드웨어적인 구현등이 연구되고 있다. 프로토콜의 처리율을 높이려면 구조가 간단해야 하지만 다양한 기능을 제공하고 융통성 있는 적용이 가능하려면 프로토콜이 복잡해진다.

그러나 앞으로의 통신망은 에러발생 확률이 낮아질 것이므로 트랜스포트 프로토콜은 고정된 일을 반복 수행하게 될 것이다. 본 논문에서는 이러한 특성을 이용하여 설계되고 제안된 MTP 프로토콜을 VHDL을 이용하여 하드웨어적으로 설계하고 Actel FPGA Chip을 이용하여 구현하였다.

MTP 프로토콜은 정보평면과 제어평면으로 구성된다. 정보평면은 에러가 발생되지 않는 한 프로토콜 상태정보에 관계없이 사용자 정보를 전달하는 고정된 일을 반복하고, 제어평면은 프로토콜 상태정보에 따라 정보평면의 동작을 제어한다. 하드웨어적으로 구현된 MTP의 처리율은 정보평면에 의해 좌우되고 700 Mbps이상이 된다.

Abstract

This dissertation deals with the design and VLSI implementation of the MTP(Multimedia Transport Protocol) protocol for the high speed networks.

High throughput, functional diversity and flexible adaptation are key requirements for the future transport protocol. However it is very difficult to satisfy all these requirements simultaneously.

Fortunately, the future networks will be very reliable. It means that the future transport protocol will usually perform some fixed functions without the protocol state information. According to this concept, we proposed and designed the MTP protocol that is consisted of Information Plane and Control Plane. Information Plane performs some fixed functions that are independent of the protocol state information as far as no error. However Control Plane manages the protocol state information and controls the operation of Information Plane.

* 동의대학교 컴퓨터공학과(jwjang@hyomin.dongeui.ac.kr)

접수일자 : 1997년 9월 1일

Our MTP protocol was finally implemented as an FPGA chip using the VHDL. We built a testbed for verification of the implemented protocol, and it was shown that the MTP protocol worked correctly and made a throughput of about 800 Mbps.

Our future works include the addition of multiplexing and multicasting capabilities to our protocol for multimedia applications.

1. Introductions

During last two decades, transmission speed in the communication networks has increased more than five orders of magnitude, however the processing speed of protocol processor has improved only 2～3 orders of magnitude. This change has shifted the bottleneck of throughput from the network to the protocol processor. In other words, we can get only Mbps throughput, even though we use Gbps networks[1,2].

To discuss protocol processing problem, let us classify protocol into three categories; low layer protocol (layer 1, 2 and 3 of OSI Reference Model-OSIRM), transport protocol (layer 3 through 6 of OSIRM), and application protocol(layer 7 of OSIRM). We usually implement low layer protocol by hardware, whose speed can be increased to Gbps orders. As transport protocol however is very complex and usually implemented by software, it becomes bottleneck to improve throughput[1, 2, 3, 4, 5].

On the other hand, multimedia application requires more powerful transport protocol. For example, transport protocol for multimedia application requires combinations of requirements such as extremely high throughput (supercomputer transmission), error-free transfer (data transmission), low delay jitter(voice conversation), fast connection (remote procedure call), and multicast (collaborative work). To support these requirements, we need a new transport protocol that can satisfy high throughput, functional diversity, and flexible adaptation[3]. In other words, multimedia transport protocol has to be able to support throughput more than Gbps order. It also has to provide all kinds of functions that are required in multimedia applications. Furthermore, it should be as flexible as to apply all kinds of applications effectively.

However, existing transport protocol can not support the requirements[3]. It is very hard to get Gbps throughput by the existing protocol such as TCP/IP. The existing protocol can not support some functions such as fast connection, multicast, and broadcast. Moreover, they are not so flexible as to apply to multimedia application effectively. For example, it is impossible to skip error control function in the existing protocols, even though it is not required in the voice conversation[2,5,6].

Multimedia transport protocol should be very complex so that it can support functional diversity and flexible adaptation. On the other side, it should be simple to get high throughput. Therefore, it is very difficult to satisfy all these requirements at once. Fortunately, probability of being unexpected event caused by error and buffer overflow, will be decreased considerably in future networks, because they will supply much more reliable service, with bit error rate as low as 10-9 compared to the networks today[2]. In addition, it is also easy to provide a large buffer because the price of semiconductor memory is decreased more and more. Therefore, transport protocol will result in performing some fixed functions without protocol state information.

From the idea, we propose a new multimedia transport protocol (hereafter, we call it MTP) in this paper. MTP is consist of two planes; information plane and control plane. Information plane performs the functions that are independent of protocol state information. However control plane manages protocol state information and controls the operation of information plane. Therefore, we can transmit user information without protocol state information, as far as there is no unexpected event.

Table 2.1 Functions of MTP

Function Name	Covering Layer in the Existion Protocols	Algorithms
Segmentation and Reassembly, Sequence Control	Transport, Low Layers	
Flow Control	Transport, Low Layers	Stop and Wait, Sliding Windows Rate Control
Acknowledgement		ACK, NAK
Error Control	Transport, Low Layers	FEC, ARQ
Error Check		Parity, CRC, LRC
Retransmission Control		Goback-n Selective Repeat
Connection Management Session Management Channel Management Call Processing	Transport, Session, Low Layers	Fast Connection Implicit Connection Broadcasting, Multicasting
Multiplexing	Transport, Low Layers	
Ecryption	Presentation Layer	DES
Failure Recovery	All Layers	
Dialogue Control	Session Layer	
Data Encoding	Presentation Layer	
Data Compression	Presentation Layer	MPEG, JPEG, X.261, MHEG
Media Synchronization		

We describe the transport function and its characteristics in chapter 2. We also give the MTP architecture and its characteristics in chapter 3. And we describe the implementation of MTP protocol in chapter 4 and evaluate the throughput in chapter 5. Finally we present conclusions in chapter 6.

2. Transport Functions and Its Characteristics

The functions of MTP can be summarized as shown in Table 2.1, where MTP is equivalent to the layer 4 through 6 of OSIRM[2,4,7].

Table 2.2 shows the characteristics of transport functions. We can classify the transport functions into three categories. The first one is information function that handles information itself. The second one is modification function that may change the content of information. The last one is state function that manages protocol state information. Information function includes SAR, sequence control, error check, multiplexing and media synchronization function. Modification function includes encryption, data encoding, and data compression function. On the other hand, state function has flow control, acknowledgment, retransmission control, connection management, failure recovery, and dialog control function. To

Table 2.2 Characteristics of Functions of MTP

Functions	Type	Time Relationsship	Implementation Characteristics	Allocation
SAR	I	Real-time		Information Plane
Sequence Control	I	Real-time	With SAR	Information Plane
Flow Congrol	S	Real-time		Control Plane
Acknowledgement	S	Real-time		Control Plane
Error Check	I	Real-time		Information Plane
Retransmission Control	S	Real-time	With Flow Control	Control Plane
Conn.Management	S	Non-realtime		Control Plane
Multiplexing	I	Real-time	Lower Part	Information Plane
Encryption	M	Real-time	Upper Part	Information Plane
Failure Recovery	S	Non-realtime		Control Plane
Dialog Control	S	Non-realtime		Control Plane
Data Encoding	M	Real-time	Upper Part	Information Plane
Data Compression	M	Real-time	Upper Part	Information Plane
Media Synchronization	I	Real-time	Upper Part	Information Plane
**) I : Information Function M : Modification Function S : Status Function				

minimize information exchange between planes, it is better to assign information function and modification function into the information plane, while state function into control plane.

We can also classify transport function as real-time function and non-real-time function, taking into account of time relationship. Connection management function is generally insensitive to processing time. However, information function and modification function should be processed with real-time. These real-time functions are most important one to affect the transport throughput.

Finally let us discuss the implementation characteristics of transport functions. As sequence control is tightly dependent on SAR, it should be implemented with SAR function. We usually implement error control and flow control function together[2]. Multiplexing function should be positioned at low part of transport layer. We can omit the function from transport layer, because bandwidth efficiency is not more important factor in future network, compared to protocol complexity. We should perform encryption, data encoding, data compression, and media synchronization at upper part of transport layer, because these functions may modify the content of information.

3. Multimedia Transport Protocol architecture

3.1 Information Plane

We use two-plane architecture in MTP, taking into account of event dependency. This means that control plane manages protocol state information but information plane should be independent of the protocol state information, as possible as it can. Therefore function of information plane should be selected according to the following guideline.

(1) Functions that are independent of the protocol state variables are allocated to information plane.
(2) To minimize the information exchange between planes, we put information function and

modification function at information plane.

Table 2.2 also shows the allocation of transport functions. We assign SAR, sequence control, and error check function to the information plane, because we can process them without protocol state information. In addition, modification function such as encryption, data encoding, and data compression function is also allocated to the information plane. On the other hand, information plane also includes multiplexing and media synchronization function to minimize the information exchange between planes.

Among the above functions, modification function should be processed before protocol overhead is involved. Therefore, modification function should be processed at top of transport layer. Media synchronization function also should be at upper part of transport layer. We can implement these functions depending on the application.

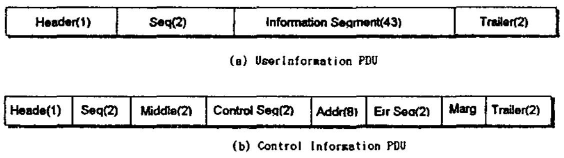


Fig. 3.1 Formats of MTP Frame

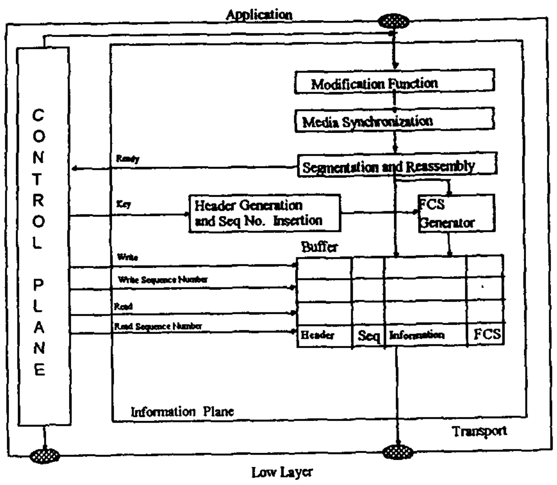
Fig 3.1 shows the format of information PDU that is processed by information plane. Information PDU has no control information so that it can be processed without protocol state information. Flag indicates delimit of PDU. We insert source and destination address into address field. These addresses are given by control plane after connection is established. Sequence number is generated by information plane itself. We use sequence number as buffer address. In other words, PDU whose sequence number is "a" will be stored at address "a". Flag, address, and sequence number are inserted just before transmitting. We then store only information field and FCS filed in buffer. Therefore, we can process the information PDU in advance, because all fields are independent of control information.

Fig 3.2(a) shows the structure of information plane of sender. We receive information from application and perform modification function and media synchronization function, if necessary. Recently many media synchronization methods are proposed[8]. We use media synchronization method that binds all media together. In other words, traffic that is given by application at same time is assigned into same PDU. Other media synchronization methods can of course be applied to MTP.

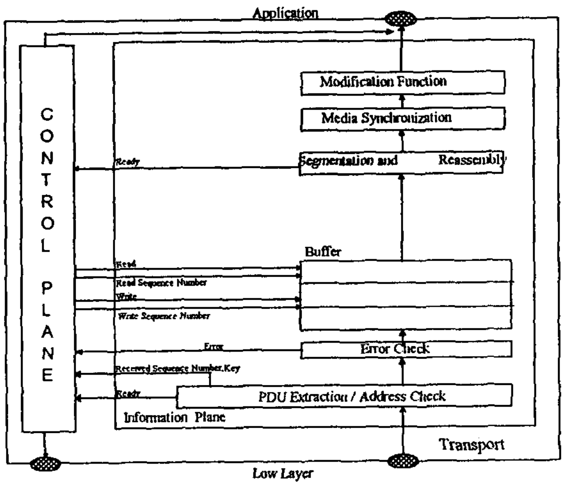
PDU processed by media synchronization module will be passed to SAR module. SAR module divides application message into information PDU and assigns sequence number to each PDU. After these procedures, we calculate FCS and store it in buffer indicated by sequence number. These procedures are able to be processed in advance, without any information from control plane. The word size of buffer is sum of information field and FCS field: While, buffer size if determined by window size of flow control algorithm. With this structure, we can read one information PDU at once.

PDU stored in buffer will be transmitted under control of control plane. In other words, control plane gives read sequence number and read signal to the information plane to control information flow. Read sequence number is address of buffer to read and indicates the sequence number of PDU to transmit. Read signal indicates the timing to transmit. Accordingly we can control information flow by read signal.

Fig 3.2(b) shows the structure of information plane of receiver. If PDU extraction/address check module detects PDU destined to itself, it gives ready signal and received sequence number to control plane. In addition, error check module reports to control plane whether it detects error or not. Control plane then checks received sequence number and error signal to know whether received



(a) Sender



(b)Receiver

Fig. 3.2 Structure of Information Plane

PDU is correct or not. If it finds no errors, it stores the PDU in buffer indicated by write sequence number. If it finds some errors, it informs sender of this fact. As we employ sequence number to buffer address, we can cover sequence control of receiver automatically.

PDU stored in buffer is passed to SAR module under control of read signal given by control plane. SAR module then reassembles PDU into a message and passes it to application after processing media synchronization and modification function, if necessary.

3.2 Control plane

Control plane manages protocol state information and controls the operation of information plane. When application requires to transfer information, control plane establishes information connection and control connection simultaneously. We use information connection to transmit user information, but we use control connection to exchange state information between transport entity. We can use signaling channel of low layer as control connection.

We also divide control plane into two modules, taking into account of event dependency. One is drive module that controls the operation of information plane directly. The other one is event module that manages protocol state information and controls the operation of drive module. The operation of drive module is also independent of protocol state information.

Fig 3.3 shows the structure of control plane of sender. Drive module of control plane generates write sequence number and write signal to receive information from application. We use WC(Write sequence Counter), FP(Free Point), and write control module to generate these signals. We store write sequence number in WC, that indicates address of buffer to write PDU. FP indicates the last address that is released by acknowledgment. Write control module is control logic to generate write signal. Therefore, buffer address indicated by WC through that indicated by FP is empty space as shown in Fig 3.3.

When write control module receives ready signal from information plane, it generates write signal to store PDU in buffer indicated by WC and increments WC by one. If WC is then equal

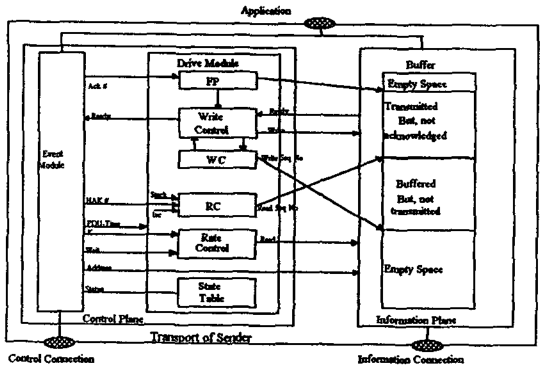


Fig. 3.3 Control Plane of Sender

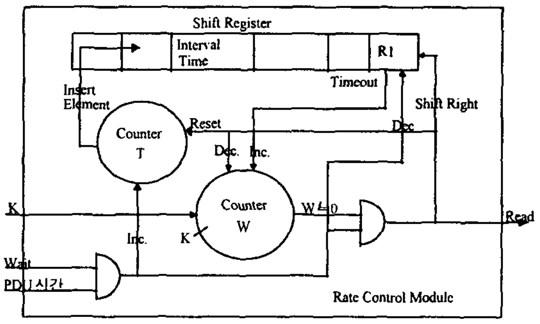


Fig. 3.4 Rate Control Module

to FP, it disables ready signal to stop information flow from application, because there is no empty space in buffer. If we can get empty space again, write control module enables ready signal to receive information from application.

On the other hand, drive module also generates read sequence number and read signal shown in Fig 3.3. We use RC(Read sequence Counter) to store read sequence number that indicates the sequence number of PDU to transmit. RC is set to 0 initially and updated whenever an information PDU is transmitted. Source of RC should be determined depending on the retransmission control algorithm. In other words, the next sequence number should be selected from three sources; increment, stack, and sequence number given by acknowledgment. If there is no error, we must select next sequence number from increment source. Otherwise we must select another source depending on retransmission control algorithm. For example, we must select the sequence number given by acknowledgment, in case of goback-n algorithm. Flow control function is controlled by read signal that is generated by rate control module. This signal is generated according to rate control algorithm based on interval time[9]. Fig 3.4 shows the rate control module. It consists of a shift register and two counters. Shift register saves interval time of the adjacent two PDU being transmitted. Counter T counts interval time and up-down counter W indicates the number of PDU to be able to transmit without acknowledgment at any instance of time.

Shift register is shifted right whenever a PDU is transmitted. The first element of shift register is decremented by one at each PDU-time(duration of one PDU) and it generates time-out signal when it equals to 0. Counter T is incremented by one at each PDU-time. It is inserted to the left end of shift register and then reset to 0 whenever a PDU is transmitted. On the other hand, counter W is initially set to K, that indicates maximum number of PDU to be able to transmit without acknowledgment. It is decremented whenever a PDU is transmitted and incremented whenever time-out occurs at shift register. We can then activate read signal at each PDU-time, as far as counter W is none-zero and wait signal is enable. We use wait signal here to stop transmitting, when receiver can not receive PDU any more.

Fig 3.5 shows control plane of receiver. It is also consist of drive module and event module. rive module generates write sequence number and write signal to receive PDU from low layer. It also generates read sequence number and read signal to transfer message to application. Write sequence number is saved in WC and read sequence number in RC.

If information plane receives PDU destined to

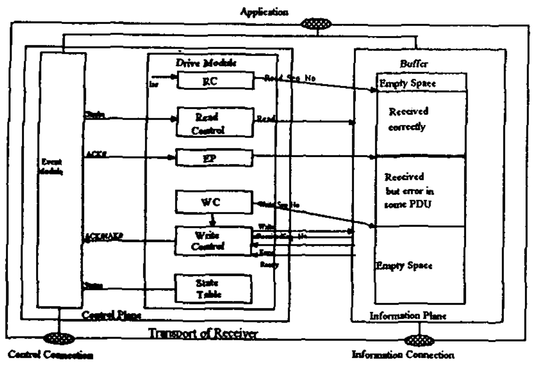


Fig. 3.5 Control Plane of Receiver

itself, it sends ready signal and received sequence number to drive module. In addition, it reports results of error detection. If it detects no error, drive module generates write signal to store received PDU into buffer and increments WC by one. If it detects error, drive module updates state table and informs the result to sender through control connection. On the other hand, as far as RC is less than EP(Error Pointer) and strobe signal is enable, we activate read signal at each PDU-time to transfer information to application and then increment RC by one. EP indicates the lowest sequence number of PDU that is received incorrectly.

As mentioned before, event module manages protocol state information and controls the operation of drive module. Fig 3.6 shows event module and its control signals. When connection is established, event module gives the initial value of drive module such as PDU-time, address, and K. PDU-time and address are determined from the call.request command given by application, while parameter K depends on characteristics of network and flow control algorithm. We can determine these signals before we start to transmit user information. Furthermore we use ready signal and strobe signal to make handshake between transport layer and application. We also use wait signal to stop transmitting in case of event such as buffer overflow. These signals will not interrupt the normal operation of drive module, as far as there is no unexpected event.

When drive module detects error, it notifies sequence number of PDU that has error, using ACK/NAK signal. Event module then makes acknowledgment to inform sender of the fact. In addition, when event module receives acknowledgment from its counterpart, it also generates ACK/NAK signal to update EP, FP, and RC register. If there are no unexpected events, event module updates only EP and FP register. We can update these registers without interrupting the normal operation of drive module, because they are not address pointer such as RC register. Therefore, we can operate drive module independently, as far as there is no unexpected event.

It is better to implement event module by software, because its functions vary with depending on the protocol state information. However, it is possible to implement information plane and drive module by hardware, because their functions are almost fixed. We can implement those functions into transport board that is compatible to system bus of general purpose computer. In this case,

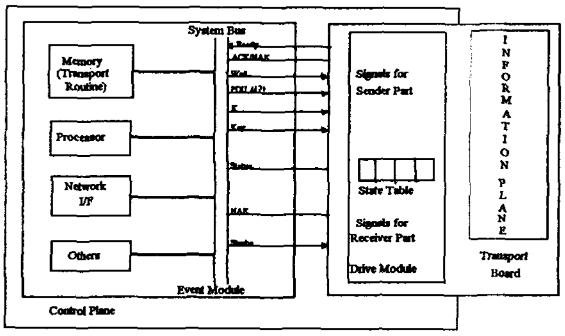


Fig. 3.6 Structure of Event Module and Control Signals

event module may be implemented by drive program to operate the transport board.

3.3 Characteristics

Finally let us talk about the characteristics of MTP. Buffering is inevitable in transport protocol because it must prepare for retransmission. Therefore maximum throughput of transport protocol is a reciprocal of buffer access time theoretically.

On MTP, information plane is responsible for transmitting user information. We can perform its function and store the result in buffer beforehand. This means that the throughput of MTP depends on only read signal shown if Fig 3.2(a) and buffer access time. We can generate read signal at each PDU-time as far as there is no unexpected event. Therefore, the throughput of MTP is reciprocal of PDU-time and can be improved to theoretical maximum. Fore example, when word size of buffer is 1 K and its access time is 100 nsec, we can get almost 10 Gbps throughput theoretically.

MTP is flexible so that we can apply it to multimedia application effectively. We can select transport function depending on the application by negotiating at connection establishment phase. For example, if we need not to control error in voice communication, we can skip error control function only by ignoring error signal shown if Fig 3.5(a). This can be achieved by software stored in event module. On the other hand, we can also change protocol control algorithm dynamically during information transfer phase, because MTP uses separate control connection.

4. Implementation

We describe about implementation of MTP protocol. At first we explain about environment and method of implementation and next we will explain production and testing of MTP protocol.

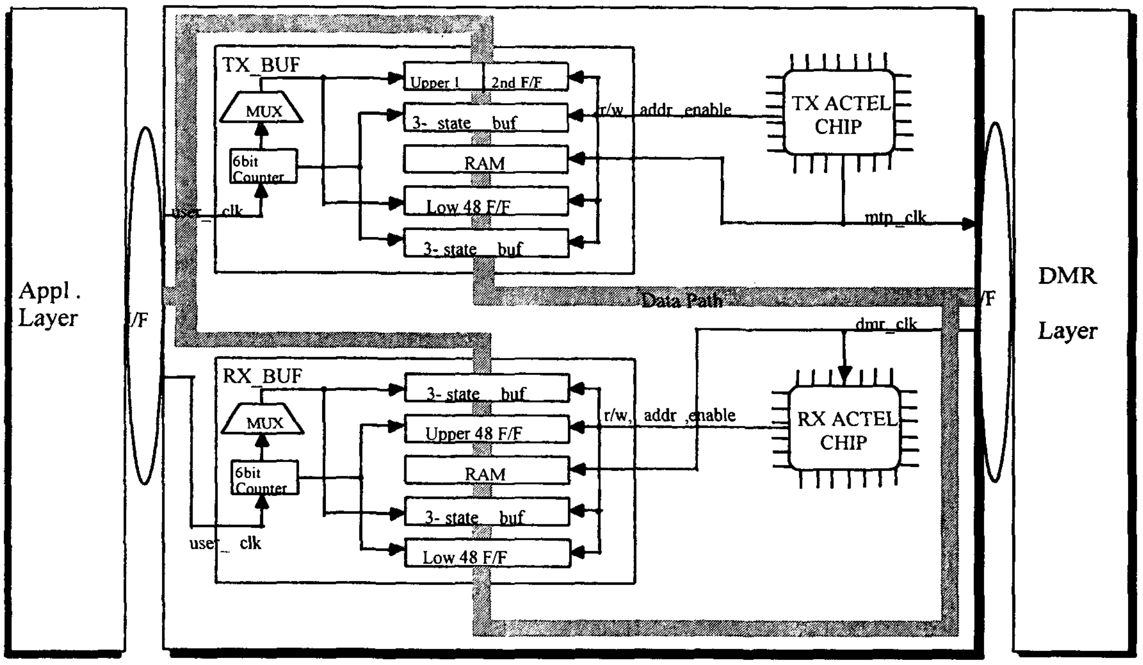


Fig 4.1 *Structure of MTP Board*

4.1 Environment and Method of Implementation

We designed MTP protocol using Powerviews VHDL which is CAD Tool and verified the correct operation using simulation function. And we made FPGA chip of MTP protocol machine using ALS which is FPGA Tool.

4.2 Production and testing

We produced MTP Board which processes MTP protocol using MTP FPGA chip which is implemented above process and SRAM. And we had tested as like test environment using DMR (Dynamic Monitor Ring) as low layer protocol[10]. Fig 4.1 shows internal structure of MTP Board including FPGA chip and commercial chip. Remainder part excluding TX_BUF at Sender Block and Remainder Part excluding RX_BUFF at Receiver Block is produced by Actel Chip(PGA Type 8000 Gate). We implemented TX_BUFF and RX_BUFF with SRAM(KM6264).

We tested MTP protocol Board using FTP service as upper layer and MAC protocol as low layer and verified the operation of test system through simple file transfer. At first user request for connection to application layer such as FTP then FTP established connection with counterpart system. After connection is established successfully, computer including FTP began to send information. On the other hand, if computer receives disconnection request form user and counterpart system then computer stop the operation and disconnect.

5. Throughput

5.1 Evaluation

We designed MTP protocol for High Speed Network. Therefore we monitored the performance at interface with Application layer.

We designed MTP protocol for implementing H/W easily as described chapter 3. Therefore Information from application layer is stored buffer as stream type by Write signal which is generated Control Plane. Information stored buffer is divided to cells and FCS is added to each cells and transfered to low layer. At that time, the processes which information is transferred to low layer and upper layer is operated parallel and it spend 70 nsec to process one cell.

Performance of MTP protocol is depend on Read signal and Buffer access time in Fig 3.5 because Buffer access time is longest among processing time of each module. In other words, it depends on Buffer cycle time generating Read signal.

Because MTP protocol transmit information a PDU per one time, so the max throughput of MTP protocol is presented as follows

$$\text{Max\_throughput} = \text{PDU\_size} * 1/\text{Cycle\_Time} \qquad (5.1)$$

Now MTP chip generate Read signal and Write signal in speed of 250 nsec by turns as presented Appendix. Namely, Buffer cycle time is 500 nsec (2 * 106 packets/sec). We got this value from Back-End Layout) simulation(Post which is previous step to chip production.

Therefore we could evaluated the throughput of MTP chip as like (5.2).

$$\text{Max\_throughput} = 1\text{cell}/500\text{nsec} = 48*8\text{bit}/500\text{nsec} = 768\text{Mbps} \qquad (5.2)$$

Now let us consider the effects of transmission errors. and we consider retransmission for correcting error which is generated randomly. Therefore we can compute the mean number of retransmissions per frame. A frame is successful if both the data and acknowledgment are correctly received. The probability of success is (1 -Pd)(1 - Pa). Therefore, the probability of failure is defined like (5.3). Table 5.1 shows definitions of symbols which was used for derivation.

Table 5.1 Definition of Symbols

Symbol	Definition
E	Bit Error Rate
Pd	Probability of a cell being error
Pa	Probability of a ACK/NAK cell being error
L	Probability that a data cell or its ACK/NAK packet is being error
R	Mean number of transmission per a cell including retransmission
F	cell length(384 bits)

L=1-(1-Pd)(1-Pa)=1-(1-E)F (5.3)

The probability that exactly k attempts are needed(i.e., k - 1 retransmission) is (1 - L)* Lk-1. This result yields an expected number of retransmission per frame of 1/(1 - L) like (5.4).

$$R = \int_0^{\infty} k(1-L)L^{k-1} dL = \frac{1}{1-L} \qquad (5.4)$$

Using this value of R, we arrive at a Throughput of MTP chip like (5.5).

Throughput= Max_Throughput/R (5.5)

Therefore when consider transmission, the throughput of MTP protocol is presents as Fig 5.1

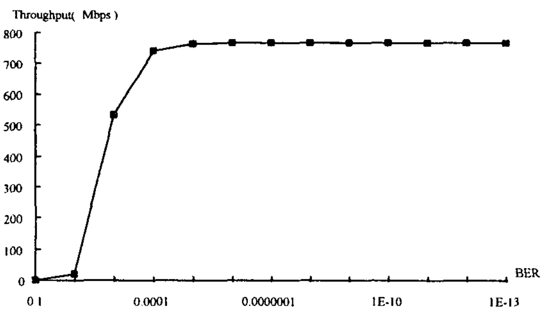


Fig 5.1 Throughput of MTP Protocol

5.2 Evaluation Analysis

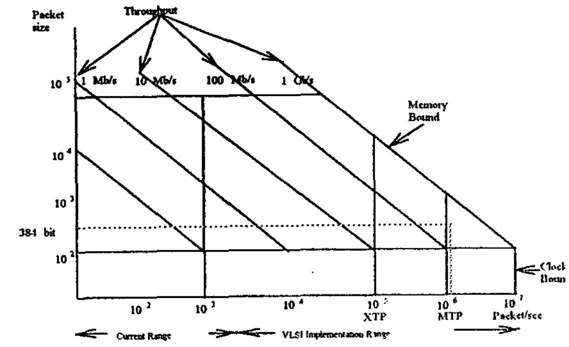


Fig 5.2 Throughput Comparison of Various Transport Protocols

It is easier to conceptualize the problem of high-performance protocol processors and the relations between the different technologies, using a speed chart as provided as Fig 5.2 below. The horizontal axis describe processing speeds as measured in packets processed per second. The vertical axis describes the size of a packet. The throughput measured in bit/second is the product of the packet size and the rates of packet processing in p/s. Since both axes use logarithmic scale, the equal throughput curves are linear. The ultimate limits on protocol processing speeds are established by the boundary lines. The vertical line on the right establish the bounds on the p/s processing rates set by chi--clock rates(and the number of cycles required to process a packet). The 1 Gbps throughput line on the right sets the bounds established by memory technology on the speeds at which packet bits may be read and written. While these physical bounds, on clock and memory access rates, may be improved slightly via high-speed logic technologies, they set the ultimate bounds on protocol processing speeds. The horizontal lines set the bounds on packet size and headers; both may be flexibly adjusted by protocol design to optimize performance goals. The goal of high-speed protocol processing may be described as increasing the processing rates measures in p/s. Current high-speed protocol implementations[11,12] ranges up to a few thousands p/s. And throughput of XTP which can be tried to implement H/W was evaluated about 105 packets/sec. But MTP protocol can be used to build RAM whose speeds range a few orders of magnitude higher, closing the gaps to the physical limits(or reaching very closely).

6. Conclusion

As speed of network becomes higher, Need for high speed multimedia transport protocol has been increased rapidly. However, it is very difficult to get high throughput by the existing protocols, because they are complex and usually implemented by software. Furthermore, they are not so flexible as we can not apply them to multimedia application effectively. Therefore, we need a new transport protocol that can support high throughput, functional diversity, and flexible adaptation.

In order to increase throughput, the protocol should be simple as possible as it can. On the other side, it should be so complex that it can support functional diversity and flexible adaptation. So, it is very difficult to satisfy all the requirements at once. Fortunately, probability of being unexpected event caused by error becomes very low in future reliable network. This implies that transport protocol will result in performing some fixed function without any protocol state information. In other words, we can formalize functions of transport protocol, taking into account of event dependency. From the idea, we propose a new high speed multimedia transport protocol(MTP).

MTP is consist of information plane and control plane. Information plane performs event-independent and fixed functions. It can operate without protocol state information, as far as there is no unexpected event. On the other hand, control plane manages protocol state information. It is consist of drive module and event module. Drive module also performs fixed functions without protocol state information. It controls the operation of information plane directly. However, event module maintains the protocol state information and handles the event caused by state change. It controls the operation of drive module but it doesn't interrupt the normal operation of drive module as far as there is no unexpected event.

Information plane performs only SAR, sequence control, and error check function for core function. We can perform these functions and store the results in buffer beforehand, because these functions are independent of protocol state information. On the other hand, flow control can be achieved by reading time of buffer. The read signal is generated by rate control module in drive module. Therefore, information plan can operate independently as far as there is no unexpected event.

Maximum throughput of MTP depends on only buffer access time. Therefore we can get more than 800 Mbps throughput by MTP. In addition, MTP is flexible and it can be implemented by hardware easily, because most functions of MTP are fixed.

References

[1] W. Doeringer, D. Dykeman, M. Kaiserswerth, B. Meister, H. Rudin, and R. williamson, "A Survey of Light-weight Transport Protocols for High-Speed Networks," IEEE Trans. on Communications, Vol. 38, No. 11, Nov. 1990, pp 2025 - 2039

[2] T. Porta and M. Schwartz, "Architectures, Featurse, and Implementation of High-Speed Transport Protocols," IEEE Network Magazine, May, 1991, pp 14 - 22.

[3] D. Schmidt, D. Box, and T. Suda, "ADAPTIVE:

A Flexible and Adaptive Transport System Architecture to Support Lightweight Protocols for Multimedia Applications on High-Speed Networks," Proceedings of 1st International Symposium on High-Performance Distributed Computing, Syracuse, NY, Sept. 1992, pp 174 - 186.

[4] Z. Haas, "A Protocol Structure for High Speed Communication over Broadband ISDN," IEEE Network Magazine, Jan. 1991, pp 64 - 70.

[5] M. Zitterbart, "High Speed Transport Components," IEEE Network Magazine, Jan. 1991, pp 54 - 63.

[6] D. Clarkm V. Jacobson, J. Romkey, and H. Salwen, An Analysis of TCP Processing Overhead, IEEE Communications Magazine, June 1989, pp 23 - 29

[7] A. Tanenbaum, Computer Networks; 2nd Edition, Prentice Hall, 1989.

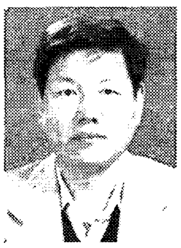
[8] P. Leydekkers and B. Teunissen, "Synchronization of Multimedia Data streams in Open Distributed Environments," PTT Research Tele- Informatics, Netherlands. pp 94 - 104.

[9] J.W. Jang and J.T. Lee, "Design and Implementation of Positive flow Controllers for High-Speed Reliable Networks," Proceedings of ICT97, Vol.3, pp. 1373-1378 , Apr. 1997.

[10] J.M. Kim, J.K. Choi, Y.H. Jeon, H.J. Jeong, and J.T. Lee, "The Design and Performance Simulation of the Dynamic Monitor Ring Protocol," Proceedingd of JCCNSS'93, pp. 113 - 118, June. 1993.

[11] V.Jacobson, UNIX 4.3 BSD TCP performance, private communication, 1988

[12] H. Inai, et.al., End-to-End Performance Modeling for Layered Communication Protocols, INFORCOM90, pp.442-449, June. 1993.



장 종 욱(將種旭)
1987. 2 부산대학교 전산통계학과 졸업
1991. 2 충남대학교 전산학과 석사
1995. 2 부산대학교 컴퓨터공학과 공학박사
1987. 2~1995. 2 한국전자통신연구원 연구원
1988. 3~현재 동의대학교 컴퓨터공학과 조교수
1996. 9~현재 한국전자통신연구원 초빙연구원