An Adaptive Universal Serial Bus (USB) Protocol for Improving the Performance of Data Communication under the Heavy Traffic

Yoon-Gu Kim*, Ki-Dong Lee**

* School of Electrical Eng. & Computer Science, Yeungnam Univ., Kyungsan, Kyungbuk, Korea
(Tel : +82-53-810-3910; E-mail: ryankim8@yumail.ac.kr)
** School of Electrical Eng. & Computer Science, Yeungnam Univ., Kyungsan, Kyungbuk, Korea
(Tel : +82-53-810-3910; E-mail: kdrhee@yu.ac.kr)

Abstract: Universal Serial Bus (USB) is one of the most popular communication interfaces. When USB is used in more extended range, especially configuring home network by connecting multiple digital devices each other, USB interface uses the bandwidth in the way of Time Division Multiplexing (TDM) so that the bottleneck of bus bandwidth can be brought under the heavy traffic. In this paper, the more effective usage of bus bandwidth to overcome this situation is introduced. Basically, in order to realize the system for transferring real-time moving picture data among digital information devices, we analyze USB transfer types and descriptors and introduce the method to enhance the detailed performance of isochronous transfer that is one of USB transfer types.

Keywords: USB, Isochronous, Bus Bandwidth, Heavy Traffic

1. INTRODUCTION

Universal Serial Bus (USB) is one of the most important developments in PC peripheral interconnect technology. It prevails that USB is adopted in new digital devices and equipments. The benefits of USB, such as ease of use, true plug and play, high performance, reduced system cost, and so on, make no doubt of the choice of USB for communication interface [1].

While USB is fast widening the range of its usage, When USB is utilized in more extended range, especially configuring home network by connecting multiple digital devices each other within a home, USB interface uses the bandwidth in the way of Time Division Multiplexing (TDM) so that the bottleneck of bus bandwidth can be brought under the heavy traffic [7]. In this paper, the more effective usage of bus bandwidth to overcome this situation is introduced. Basically, in order to realize the system for transferring real-time moving picture data among digital information appliances, we analyze USB transfer types and descriptors, data structures with the defined format for configuration information exchange between a host and a device, and introduce the method to enhance the detailed performance of isochronous transfer that is one of USB transfer types.

In the case that a configuration descriptor of a USB device has an interface descriptor that has two alternate settings, if the isochronous transfers between host and devices are not processed smoothly due to excessive bus traffic, the application software of the device requires a new configuration through Set_Interface request that is one of the predefined requests in the USB specification and changes the other one of both alternate settings in the interface descriptor. As the result of this adaptive configuration, the least data frame rate for the real-time moving pictures is guaranteed to a device that the sufficient bandwidth is not allotted continuously in the heavy traffic situation. And if the bus traffic becomes normal, the algorithm for returning to the original alternate setting is also introduced. This introduced method resolves the bottleneck of moving picture transfer that can be occurred in home network interconnected by multiple digital devices [5].

2. BACKGROUND

USB System is composed of main four blocks that are client software/USB driver (USBD), Host Controller Driver (HCD), Host Controller (HC), USB device and works by interfacing each other. HCD and HC are working for data transfer between client software and USB devices [1-4]. Client software generates and receives function-specific data to/from a function endpoint of USB device via calls and callbacks requesting I/O Request Packets (IRPs) with the USBD interface. USBD converts data in client IRPs to/from device endpoint via calls and callbacks with the appropriate HCD. HCD converts IRPs to/from transactions and organizes them for manipulation by the HC. HC takes transactions and generates bus activity via packets to move function-specific data across the bus for each transaction. The procedure of USB information conversion from client software to bus is shown in Fig. 1.

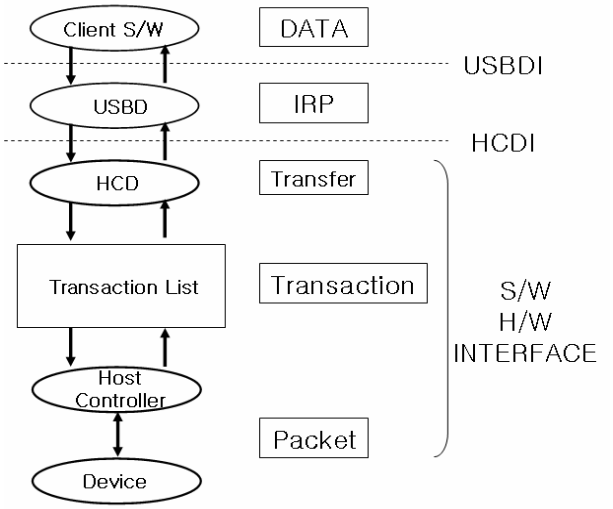

Fig. 1 USB information conversion from client s/w to bus

2.1 USB Transfer Types

There are four USB transfer types; Control, Isochronous,

Interrupt, Bulk. Each transfer type is possible to configure optionally according to the service requirement and condition between client software and USB device. Each transfer type has its own transfer features.

Control transfer is bi-directional and support configuration, command or status communications between the host and the function of device. A control transfer consists of 2 or 3 stages, a setup stage, data stage and status stage. A data stage may or may not exist according to request command type. In the setup stage, the host sends a request to a function of device. In the data stage, data transfer occurs in the direction as indicated in the setup stage (IN: device to host, OUT: host to device). And in the status stage, the function returns a handshake to the host.

Isochronous transfer is uni-directional and transmits data periodically with a. But isochronous transfer is not sensitive to delivery failure due to errors so that no retry is required. The maximum transfer available using isochronous transfer is 8.814Mbps and the maximum packet size for isochronous transfer 1023 bytes per frame of 1ms.

Interrupt transfer is also uni-directional and only inputs to the host. For full speed devices, the endpoint can specify the polling period from 1ms to 255ms. It is used to support data transfers that are small in transmission size and high in transmission frequency.

Bulk transfer is used to support the case that communicating large amounts of data accurately is more important and the time of delivery is not critical. Whenever the unused bandwidth of bus is available, the transfer is possible without periodicity. In the case of delivery failure due to errors, the transfer is retried.

2.2 Descriptor Types

Descriptors are data structures, or formatted blocks of information, that enable the host to learn about a device. All USB peripherals must respond to the host's requests for the USB descriptors. There are device, configuration, interface, endpoint and string descriptor in the type of standard descriptors and they have each level from higher to lower in sequence. The higher-level descriptors inform the host of the information of any additional, lower-level descriptors and can have one or more lower-level descriptors according to the supporting features [2].

First of all, the device descriptor is the first information the host reads on device attachment. It has basic information about the device and especially has a unique Vendor ID and a Product ID that identify the device among many devices.

The configuration descriptor describes the device's features and abilities. It contains information about the device's use of power and the number of interfaces supported. When the host requires a configuration descriptor (Get_Desriptor), the device returns information of all subordinate descriptors that the configuration descriptor has, including one or more interface descriptors and optional endpoint descriptors.

The interface descriptor describes information about a specific interface that means a set of endpoints used by a device feature or function in a configuration. After the device is configured by a configuration descriptor, the interface descriptor has a field for an alternate setting, or bAlternateSetting, in order to change the endpoints and their properties according to the status of device. This paper utilizes these features of USB configuration.

The endpoint descriptor is specified for an interface and describes information about address, transfer type and bandwidth of each endpoint.

The string descriptor is optional and contains descriptive text information to string reference pointer in each descriptor [6].

3. PROPOSED MODEL AND PERFORMANCE ANALYSIS

Here we propose and simulate a USB bandwidth usage model for the enhancement of transfer performance and analyze and evaluate the result of simulation.

3.1 Proposed Model

Basically, as shown in the Fig. 2, we assume that the device descriptor of a USB device has two configuration descriptors and one of the two ones, Configuratioin0, has two interface descriptors. Then let's set that the first interface descriptor has two alternate settings that define the different features of endpoints, such as transfer type, maximum transfer packet size, and interval. According which bAlternateSetting, a field of the interface descriptor, is selected, the interface feature for the device function is changed and worked with a new interface setting. The host reads the alternate setting of current interface with Get_Interface request and requests an alternate interface through Set_Interface request at the time to require the change of the alternate setting for the more suitable USB bandwidth usage under the heavy traffic. In order to explain these situations in detail, we configure two alternate settings of endpoints for the bAlternateSetting of interface descriptor as shown in Table 1 and Table 2.

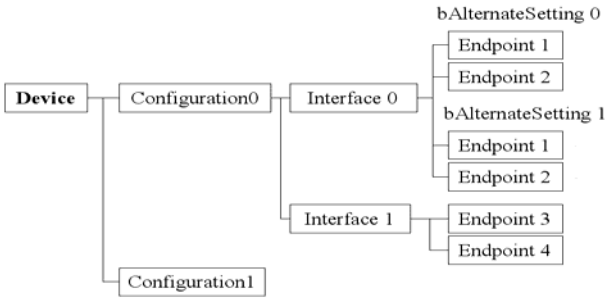

Fig. 2 Alternate settings for interface descriptor

In the case of bAlternateSetting = 0, Table 1 lists the important fields of endpoints setting.

Table 1 Endpoints Setting of bAlternateSetting = 0

Endpt No	Field	Value	Note
Endpoint 1	bmAttributes	01	Isochronous
	wMaxPacketSize	1023	Byte
Endpoint 2	bmAttributes	11	Interrupt
	wMaxPacketSize	64	Byte

In the case of bAlternateSetting = 1, Table 2 lists the important fields of endpoints setting.

Table 2 Endpoints Setting of bAlternateSetting = 1

Endpt No	Field	Value	Note
Endpoint 1	bmAttributes	01	Isochronous
	wMaxPacketSize	512	Byte
Endpoint 2	bmAttributes	11	Interrupt
	wMaxPacketSize	64	Byte

In the situation that multiple devices are interconnected and communicate with the host, each device utilizes a part of total

bandwidth allotted from the host in the way of TDM. At this time, a specific device that has a configuration based on the above configuration and each descriptor's description information is attached to the host additionally. When sufficient bandwidth is not allotted to the device and transfer requests of the device are delayed continuously, the client software retains the transfer delay count (nDelayCount) and checks whether the count exceeds a threshold count (nThresholdCount). If so, the client software of the device requests changing a bAlternateSetting of the interface descriptor. By the request, the host sends Set_Interface request for change of bAlternateSetting value in the interface descriptor and the device acknowledges the request. As the result, the isochronous transfer request between the host and device transmits by a new wMaxPackitSize in the bAlternateSetting 1 that is a half of wMaxPacketSize in the bAlternateSetting 0. This procedure enables more adaptive and effective usage of bandwidth according to the current usage status of USB bandwidth to the device that is not allotted the transfer bandwidth for a specific transfer delay count among devices competing to secure the available bandwidth.

The procedure to change an alternate setting for the interface descriptor is shown in Fig. 3. A device obtained the priority in the bandwidth competition is first allotted the bandwidth for the specified transfer and the rest of bandwidth is allotted to the other device that requests its transfer bandwidth. If the rest of USB bandwidth is less than the bandwidth that the other device requests, the communication of a frame is completed without using the rest of USB bandwidth. That results in the extravagance of USB bandwidth and reduction of total data transmission quantity and rate. Therefore, this paper introduced a model to enhance transmission performance of USB that each interface descriptor has additional configuration information to adapt to the rest of bandwidth according to the current status of USB bandwidth and transfer requests of the device retry in the changed configuration in case of continuous transfer delays.

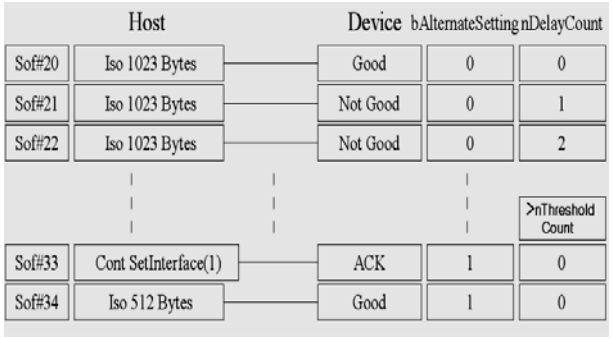

Fig. 3 Procedure to change an alternate setting of Interface Descriptor

Figure 4 shows a procedure to return to the normal setting in the case that USB bus bandwidth becomes sufficient for the transfer. In order to return to the normal setting, client software requests a IRP that has two transactions of wMaxPacketSize supported in the changed configuration and checks whether the two transactions is completed without any transmission delay. If the response to IRP is case A of Fig. 3, client software keeps the changed configuration. If it is case B, client software requests to return to the original configuration. These procedures showed more enhanced performance in total data reception quantity and rate in the simulation result of this paper.

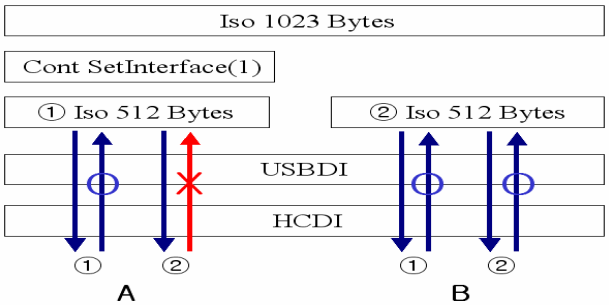

Fig. 4 Return to the normal setting

3.2 Performance Analysis

Figure 5 shows the execution screen of simulation for the adaptive model proposed in section 3.1. This simulation was programmed to simulate 3 types of situations. The first one is a case that the total of each transfer packet size exceeds the maximum USB bandwidth, Total Size > 1500bytes/ms (12 Mbps), and each device becomes to compete for the USB bandwidth to secure bus time for its transfer. The second one is a case that the total packet size is same or less than the maximum bandwidth, Total Size <= 1500bytes/ms (12 Mbps), and the needed bandwidth for all transfer requests can be allotted in a frame. The third one is a case to exceed the maximum bandwidth similarly with the first one. But the difference is that the third case applies the proposed model to this simulation and was programmed to simulate an adaptive situation that a device can vary the maximum packet size (wMaxPacketSize) adapting itself to the current bandwidth status.

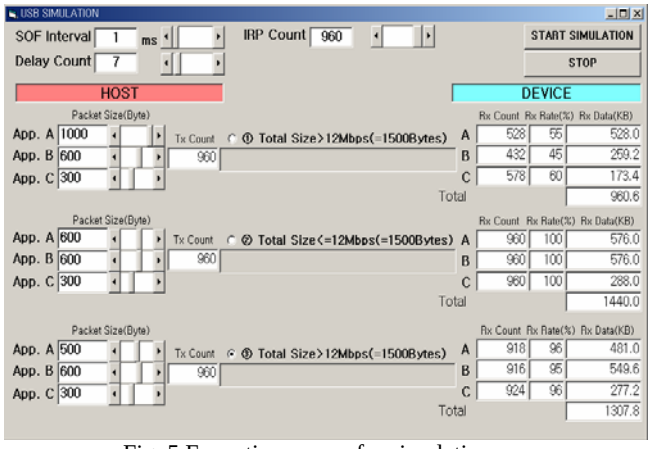

Fig. 5 Execution screen for simulation

The simulation made 10 times of experiment for each situation with SOF (Start Of Frame) per 1ms and 960 counts of IRPs in a network environment that 3 types of devices are interconnected to the host. The performance analysis based on the simulation result is shown in Fig. 6 and Fig. 7. As shown in Fig. 6, the third case of simulation that enables an adaptive usage of bandwidth depending on the current status can receive 33%(954.3:1267.8 = 1:1.328) more in the data reception quantity than the first case that competes for the bandwidth security. These results can be more or less differential according to the setting of data packet size for the simulation but the adaptive bandwidth usage shows more data reception quantity on the average than the competitive one.

Figure 7 shows data packet reception rate of each situation. This illustrates that the competitive situation cannot transfer

data packets in case of the failure to secure sufficient bandwidth, while the adaptive situation tries to transfer data packets adapting to the current bandwidth status in order to make the best of the maximum USB bandwidth. The mean data reception rate of more than 95% means that a device can have more chances to receive data packets consecutively but with the adapted packet size so that can reduce or resolve some delay phenomenon of the real-time moving picture's reproduction and enhance the quality of reproduction.

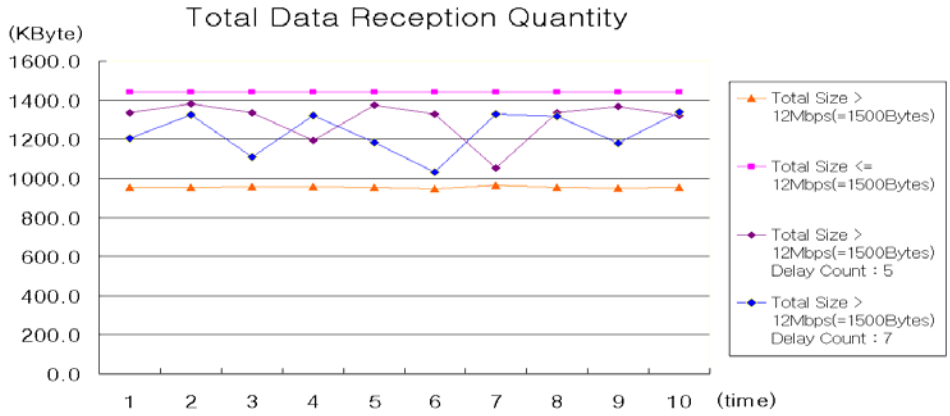

Fig. 6 Comparison of data packet reception quantity

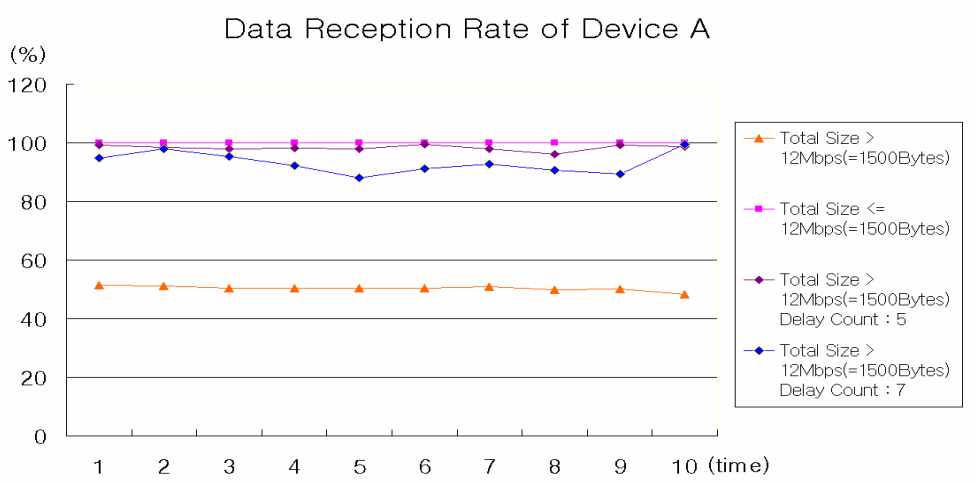

Fig. 7 Comparison of data packet reception rate

4. CONCLUSION

In this paper, we proposed a model for more efficient usage of USB communication bandwidth under the heavy traffic. The simulation result of the proposed model showed that the adaptive situation for the bandwidth results in more enhanced data reception quantity and rate in the real communication than the competitive situation. USB is widely used as one of the most popular and easy communication interfaces and estimated that the range of usage will be more and more extended and applied in various types. Especially, in the case of the moving picture transmission of multiple interfaces or the multimedia environment, when USB is adopted as the communication interface, the efficient usage of limited USB bandwidth will be the most important issue. Therefore, if the model proposed in this paper is applied to these situations, we can anticipate the enhancement of interface performance among devices.

For the more suitable enhancement of transfer performance, it is needed to develop an algorithm to determine the optimum threshold value of transmission delay count (nThresholdCount) and to be applied to the proposed model.

REFERENCES

[1] Wooi Ming Tan, “Developing USB PC Peripherals Second Edition Using the Intel 8x930Ax USB Microcontroller”, *Annabooks*, 1999.

[2] Jan Axelson, “USB Complete Everything You Need to Develop Custom USB Peripherals” Second Edition, *Lakeview Research*, 2001.

[3] John Hyde, “USB Design by Example A Practical Guide to Building I/O Devices”, *Intel Press*, 2001.

[4] Compaq, Intel, Microsoft, NEC, “Universal Serial Bus Specification Revision 1.1”, 1998.

[5] Se-Il Jun, Doo-Bok Lee, “Development of Data Transfer Program Using USB Interface”,*Korea Information Processing Society*, Vol.7, No.5, pp. 1553-1558, 2000.

[6] Kim Hyung Hun, “USB GUIDE Universal Serial Bus”, *Ohm*, 2002.

[7] Chiewon Lee, Jongwook Jang, E.K. Park, Sam Makki, “An Analysis of the Performance of TCP over IEEE 1394 Home Networks”, *IEEE*, 1999.