USB(Universal Serial Bus)의 데이터 송수신 성능향상을 위한 적응성 통신방식

An Adaptive USB(Universal Serial Bus) Protocol for Improving the Performance to Transmit/Receive Data

김윤구, 이기동
영남대학교 컴퓨터공학과

Kim Yoon-Gu, Lee Ki-Dong
Dept. Computer Engineering, YeungNam University

요약

USB(Universal Serial Bus)는 최근 컴퓨터시스템에서 가장 인기 있는 인터페이스 중의 하나인데 이를 보다 확대하여 사용 시 즉, 다수의 디바이스들이 서로 연결되는 댁내망을 구성할 때는 버스대역을 시분할 다중화(TDM, Time Division Multiplexing) 방식으로 사용하는 USB의 특징상 버스대역의 병목현상을 초래하게 되는 데 이러한 상황을 보다 효율적으로 대처할 수 있는 버스 대역 사용방안을 제안 한다. 기본적으로 USB 통신방식으로 디지털 정보가전 간 실시간 동영상 정보 전송을 위한 시스템을 구현함에 있어 보다 효율적인 데이터 전송을 위한 USB 데이터 전송타입과 디바이스 간 정보 교환을 위한 정의된 포맷을 가지는 데이터 구조인 각 디스크립터(Descriptor)들을 분석하여 동영상 전송을 위한 등시성(Isochronous)전송 방식의 세부 성능 향상 방법을 제안한다. 하나의 디바이스 설정(Configuration)이 2개의 대안적 설정(AlternateSetting)을 가지는 인터페이스(Interface)를 가정할 경우 버스 트래픽 과다에 의한 Isochronous 전송이 원활하지 않을 경우에 인터페이스의 AlternateSetting을 변경하고 설정요청(SetInterface() Request)하여 충분한 버스대역을 할당받지 못 하는 디바이스에 있어서도 최소한의 데이터 프레임률을 보장하고자 한다. 그리고 버스 트래픽이 정상인 경우 원래의 AlternateSetting으로 복귀하는 알고리즘을 제안한다. 이 제안 방법으로 다수의 디바이스들이 연결되어지는 홈 네트워크 댁내망에서의 동영상 데이터 전송 시 발생할 수 있는 병목현상을 해결한다.

Abstract

USB(Universal Serial Bus) is one of the most popular communication interfaces. When USB is used in an extended range, especially configurating In-home network by connecting multiple digital devices each other, USB interface uses the bandwidth in the way of TDM(Time Division Multiplexing) so that the bottleneck of bus bandwidth can be brought. 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 realtime moving picture data among digital information devices, we analyze USB transfer types and Descriptors and introduce the method to upgrade detailed performance of Isochronous transfer that is one of USB transfer types.
In the case that Configuration descriptor of a device has Interface descriptor that has two AlternateSetting, if Isochronous transfers are not processed smoothly due to excessive bus traffic, the application of the device changes AlternateSetting of the Interface descriptor and requires a new configuration by SetInterface() request. As a result of this adaptive configuration, the least data frame rate is guaranteed to a device that the sufficient bandwidth is not alloted. And if the bus traffic is normal, the algorithm to return to the original AlteranteSetting is introduced. this introduced method resolve the bottleneck of moving picture transfer that can occur in home network connected by multiple digital devices.

I. 서 론

USB는 최근 컴퓨터시스템에서 PC 주변기기 간 상호연결 기술에 있어서 가장 중요한 발전 중의 하나이다. 많은 수의 새로운 디지털 장비 또는 장치에 있어 USB를 통신 인터페이스로 채택하는 것이 보편화되고 있고 그 사용의 편의성, 진정한 Plug and Play, 높은 성능, 절감된 시스템 비용 등의 장점들이 USB의 선택을 의심치 않게 하는 이유들이다.[1] 이렇게 빠르게 그 사용 분야가 넓혀가고 있는 가운데 USB 적용 부분을 보다 확대하여 사용시 즉, 다수의 디바이스들이 서로 연결되는 홈 네트워크 댁내망을 구성할 때는 버스대역을 시분할 다중화(TDM, Time Division Multiplexing)방식으로 사용하는 USB의 특징상 버스대역의 병목현상을 초래하게 되는 데 이러한 상황을 보다 효율적으로 대처할 수 있는 버스 대역 사용 방안을 본 논문에서 제안한다.

기본적으로 USB 통신방식으로 디지털 정보가전 간 실시간 동영상 정보 전송을 위한 시스템을 구현함에 있어 보다 효율적인 데이터 전송을 위한 USB 데이터 전송타입과 디바이스 간 정보 교환을 위한 정의된 포맷을 가지는 데이터 구조인 각 디스크립터(Descriptor)들을 분석하여 동영상 전송을 위한 등시성(Isochronous) 전송 방식의 세부 성능향상 방법을 제안한다. 하나의 디바이스 설정(Configuration)이 2개의 대안적 설정(AlternateSetting)을 가지는 인터페이스를 가정할 경우 버스 트래픽 과다에 의한 Isochronous 전송이 원활하지 않을 경우에 Interface의 대안적 설정을 변경하고 설정변경요청(SetInterface() Request)하여 충분한 버스대역을 할당받지 못 하는 디바이스에 있어서도 최소한의 데이터 재생 프레임률을 보장하고자 한다. 그리고 버스 트래픽이 정상인 경우 원래의 설정으로 복귀하는 알고리즘을 제안한다. 이 제안 방법으로 다수의 디바이스들이 연결되어지는 홈 네트워크 댁내망에서의 동영상 데이터 전송 시 발생할 수 있는 병목현상을 해결한다.[5]

II. 관련 연구

USB 시스템은 클라이언트 소프트웨어/USB 드라이브, 호스트 컨트롤러 드라이브(HCD), 호스트 컨트롤러(HC), USB 디바이스의 주된 4개의 블록이 각 블록 간 인터페이스로 이루어진다. 호스트 컨트롤러 드라이버와 호스트 컨트롤러가 클라이언트 소프트웨어와 USB 디바이스 사이에서 데이터의 전송을 위해 동작한다.[1]-[4]

1. USB 전송타입

USB 전송방식으로는 제어 전송(Control), 등시성 전송(Isochronous), 인터럽트 전송(Interrupt), 벌크 전송(Bulk)이 정의되어 있다. 각 전송방식은 클라이언트 소프트웨어와 USB 디바이스 간의 서비스 요구 특성과 조건에 따라 선택적으로 설정이 가능하다.

각 전송방식의 특징적 내용은 아래와 같다.

■ 제어 전송(Control)

제어 전송은 양방향성이며 호스트와 디바이스의 기능(Function) 간의 설정, 명령, 상태정보 전송 역할을 지원한다. 제어전송은 2단계 또는 3단계의 과정, 즉 Setup 단계, Data 단계(선택적), Status 단계를 거치며 전송이 이루어진다. Setup 단계에서 디바이스의 Function에 하나의 요청(Request)을 보내고, Data 단계에서 Setup단계에서 지시된 방향(IN: 디바이스에서 호스트로, OUT: 호스트에서 디바이스로)으로 데이터를 보낸다. 그리고 Status 단계에서는 호스트와 디바이스 간 데이터 송수신의 상태결과를 알려준다(Handshake).

■ 등시성 전송(Isochronous)

등시성 전송은 단방향 또는 양방향성이며 일정한 전송률을 보장하며 데이터를 주기적으로 전송하지만 전송에러에는 민감하지 않아 재전송 등의 요구를 하지 않는 것이 특징이다. 최대 전송속도는 USB 1.1의

경우 1ms의 프레임당 1023 Bytes로 초당 최대 8.184 Mbps 전송이 가능하다.

■ 인터럽트 전송(Interrupt)

인터럽트 전송은 단방향성이며 호스트로의 입력만 가능하다. 1ms에서 255ms까지의 폴링주기를 가지며 작은 데이터 전송량과 높은 데이터 전송빈도에 유리한 전송 방식이다.

■ 벌크 전송(Bulk)

Bulk 전송은 데이터 전송 시간적인 측면 보다는 많은 양의 정보를 보다 정확하게 전송하는 데 그 중요성을 두며 버스 대역폭의 여유가 있을 시 비주기적으로 전송을 하게 된다.

2. 디스크립터(Descriptor) 타입 분석

디스크립터는 호스트가 접속되는 디바이스에 대해 그 디바이스의 특성을 알 수 있는 정보를 USB에서 정의하는 포맷의 데이터 구조이다. 모든 USB 주변기기들은 호스트의 표준 USB 디스크립터 요청에 일정한 포맷으로 응답을 해야 한다. 그 종류에는 Device, Configuration, Interface, Endpoint, String 등이 있으며 그 순서대로 상위에서 하위로의 레벨이 정해진다. 상위레벨 디스크립터r는 추가적인 하위레벨 디스크립터의 정보를 호스트에 알려주게 되며 지원하는 특성에 따라 복수개의 하위레벨 디스크립터를 가질 수 있다.

- 우선적으로 Device 디스크립터는 디바이스 접속 시 호스트가 첫 번째로 알게 되는 디바이스 정보로 USB 디바이스에 대한 일반적인 정보를 가지며 디바이스 간 구분을 위한 고유한 ID 정보를 가진다.
- Configuration 디스크립터는 특정 디바이스의 설정에 대한 정보를 기술하고 있으며 설정 내에 현재 설정값과 그 하위레벨 디스크립터인 Interface 디스크립터의 수 등을 알려준다. 호스트가 Configuration 디스크립터를 요청(Get_Descriptor)시 이 설정에 관련된 모든 Interface와 Endpoint 디스크립터 정보를 함께 반환한다.
- Interface 디스크립터는 하나의 Configuration 내에 있는 특정 Interface에 관한 정보를 기술하고 있다. Interface는 디바이스가 설정이 된 이후에 그 Endpoint들과 그 특성에 변화를 줄 수 있도록 하기위해 대안적 설정(Alternate Setting)이 가능한 필드를 두고 있다. 본 논문에서는 이러한 USB의 특징들을 이용한다.
- Endpoint 디스크립터는 하나의 Interface를 위해 사용되는 각 Endpoint가 가지는 디스크립터로 각 Endpoint의 주소 와 대역폭 정보를 기술한다.
- String 디스크립터는 선택적이며 각 디스크립터 내의 String 참조 포인트에 대한 문자정보를 기술한다.[6]

III. 제안 모델 및 성능 분석

1. 제안 모델

기본적으로 아래의 그림 1과 같이 하나의 디바이스 디스크립터에 2개의 설정 디스크립터가 있고 그 하나의 설정 디스크립터, 즉 Configuration0 에 2개의 인터페이스 디스크립터가 있다고 설정하고 그 하나의 인터페이스의 Function을 위한 Endpoint들에 대해 대안적 설정(AlternateSetting)이 2개 있다고 설정하자. 여기서 Interface 디스크립터 설정필드인 bAlternateSetting에 대안적 설정의 값을 설정함에 따라 Interface의 기능(Function) 특징이 변경되어 동작한다.

그림 1에서 Interface 디스크립터의 설정필드가

- bAlternateSetting=0인 경우 표1과 같이 Endpoint를 설정한다.

[표 1] bAlternateSetting = 0인 Endpoint 설정

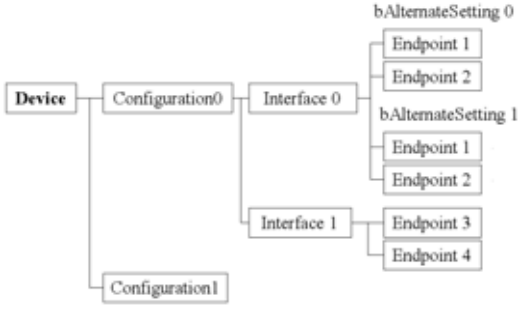
Endpt No	필드	값	비고
Endpoint 1	bmAttributes	01	Isochronous
	wMaxPacketSize	1023	Byte
Endpoint 2	bmAttributes	11	Interrupt
	wMaxPacketSize	64	Byte

• bAlternateSetting = 1인 경우 표2와 같이 Endpoint를 설정한다.

[표 2] bAlternateSetting = 1 인 Endpoint 설정

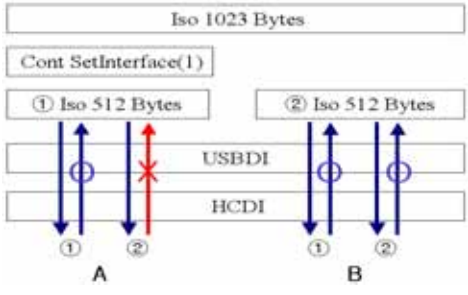
Endpt No	필드	값	비고
Endpoint 1	bmAttributes	01	Isochronous
	wMaxPacketSize	512	Byte
Endpoint 2	bmAttributes	11	Interrrupt
	wMaxPacketSize	64	Byte

아래의 그림은 위의 설정사항을 도식화한 내용이다.



▶▶ 그림 1. Interface Descriptor의 대안적 설정

위의 설정정보와 각 디스크립터들의 기술정보를 토대로 여러 대의 디바이스들의 USB 통신방식으로 서로 통신을 하는 경우 USB의 제한적인 통신대역을 시분할 다중화 방식으로 대역을 할당받아 사용하게 된다. 이 때 특정 디바이스가 충분한 대역폭을 할당받지 못 하여 전송요청이 계속 지연될 경우 전송지연회수를 카운트하고 기억하고 있다가 그 전송지연회수가 일정 회수를 초과하는 경우에 그 디바이스의 어플리케이션S/W에서 설정(Configuration) 변경을 요청하도록 한다. 이 요청으로 Interface 디스크립터의 bAlternateSetting 필드를 변경하게 된다. 즉 최초 설정보다 한 단계 낮은 수준의 bAlternateSetting으로 변경요청(SetInterface( ) Request 사용)하고 변경승인을 얻으면 그 설정을 기준으로 전송을 다시 요청하게 된다. 이렇게 하면 특정 디바이스의 전송 요청이 최초 설정 bAlternateSetting 0의 wMax-PackitSize 1/2로 줄어든 wMaxPackitSize로 이루어지게 된다. 이 과정을 통해 USB 대역폭 확보를 위해 경쟁을 하던 디바이스 중 일정 전송지연회수 동안 전송 대역폭을 할당 받지 못한 디바이스에게 현재 USB 대역폭 상태에 따라 설정정보에 변화를 줄 수 있도록 하여 현재 대역폭을 보다 효율적으로 사용하도록 한다. 구체적으로 대역폭 경쟁에서 우선권을 가진 디바이스가 대역폭을 할당받고 그 나머지 대역은 다른 디바이스의 최대전송크기(wMaxPacketSize) 조건에 맞지 않아 그대로 사용하지 않은 상태로 전송을 마무리한다면 그 나머지 대역에 대한 낭비가 발생하게 되고 전체적인 데이터 전송량도 줄어들게 된다. 이에 그 나머지 대역폭에 대해 적응할 수 있는 설정정보를 가지고 있다가 대역폭을 할당 받지 못한 상황이 계속 되는 경우에 있어서는 설정을 바꾸어 현재 대역폭의 상황에 적응해가면서 전송을 하게 된다.



▶▶ 그림 2. 정상 설정으로의 복귀

그리고 그림 2에서는 USB 버스 대역폭의 정상인 경우 원래의 설정으로 복귀하기 위해 설정 변경 후의 wMaxPacketSize로 2개의 트랜잭션을 가지는 IRP

를 요청하여 2개의 트랜잭션이 모두 전송되었음을 통보받는 경우에 원래의 설정으로 복귀하는 과정을 도식화하였다.

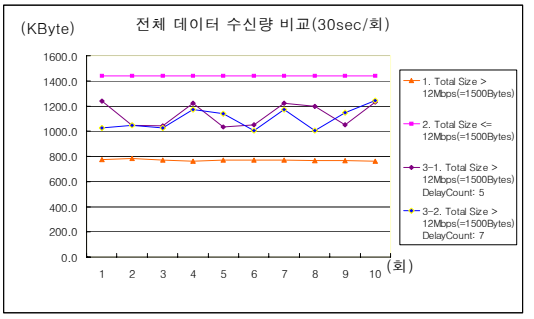
이러한 과정이 본 논문에서 실험치 결과로 기존 대역폭 경쟁의 경우 보다 전체 데이터 수신률과 데이터 수신량에 있어 보다 향상된 성능을 보이고 있음을 확인할 수 있다.

2. 성능 분석

앞 절의 제안 모델에 대한 시뮬레이션 실행화면을 그림 3에서 보여주고 있다. 시뮬레이션에서는 3 가지 경우를 선택적으로 실험할 수 있도록 프로그램 하였는데 첫 번째로 각 디바이스의 전송 패킷 크기의 합이 대역폭을 초과하는 경우(Total Size > 12Mbps (1500bytes/ms))의 실험으로 대역폭의 할당을 서로 경쟁하도록 하였고 두 번째는 전송패킷의 합이 대역폭이내 인 경우(Total Size <= 12Mbps (1500bytes /ms))의 실험으로 전체 전송크기가 대역폭 이내이므로 정상적으로 대역폭할당이 모두 이루어지는 경우이다. 세 번째로는 첫 번째와 마찬가지로 전체 전송패킷 크기가 대역폭을 초과하는 경우인데 다른 점은 앞서 제안된 모델을 적용하여 대역폭의 현재 상황에 적응해 가면서 전송패킷크기를 가변적으로 설정할 수 있도록 프로그램 하여 시뮬레이션 하였다.

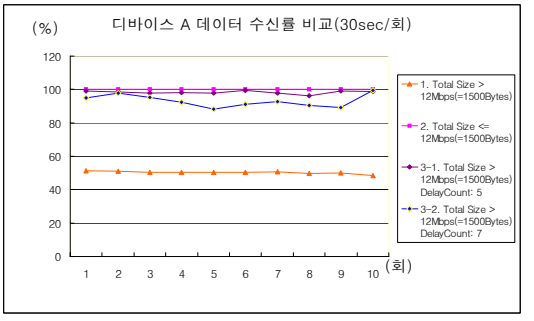
▶▶ 그림 3. 시뮬레이션 실행 화면

시뮬레이션은 1ms의 SOF(Start Of Frame) 간격과 실험 Run Time을 30Sec로 하여 각 경우에 대해 10회에 걸쳐 반복 수행하였고 그 결과를 토대로 한 성능 분석을 아래의 데이터 차트로 설명해 주고 있다.



▶▶ 그림 4. 시뮬레이션 결과I. 데이터 수신량 비교

그림 4에서 보는 결과는 대역폭을 경쟁하게 되는 첫 번째 경우 보다 대역폭의 현재 상황에 따라 적응성 있게 대역폭을 사용하는 세 번째 경우가 전체 데이터 수신량을 비교해 볼 때 평균 42%의 데이터량을 더 수신할 수 있는 결과를 보여주고 있다. 이 결과치는 시뮬레이션시 설정 데이터량에 따라 달라질 수는 있으나 평균적으로 대역폭 경쟁의 경우보다 대역폭 적응성의 경우 평균적으로 높은 데이터 수신량을 보이고 있다.



▶▶ 그림 5. 시뮬레이션 결과II. 데이터 수신률 비교

그림 5에서는 각 디바이스에서의 데이터 패킷 수신률을 나타내고 있다. 이는 대역폭 경쟁의 경우 충분한 대역폭의 여유가 없을시 전송을 하지 않게 되므로

수신률이 전체적으로 떨어지지만 대역폭 적응성의 경우엔 현재 대역폭의 상황에 적응해 전체 대역폭에 맞는 패킷 전송을 시도하므로 수신률은 상대적으로 높게 나타나고 있음을 알 수 있다. 평균 95%이상의 데이터 수신률이 의미하는 바는 연속적으로 데이터를 수신할 수 있어 하나의 디바이스에서 계속 대역폭을 할당받지 못 하는 경우에 발생하는 디바이스에서의 동영상 재생프레임이 끊기는 현상을 현저하게 줄이고 해결할 수 있다는 의미를 가진다.

IV. 결 론

본 논문에서 USB의 특성상 제한된 대역폭을 사용함에 있어 다수의 디바이스들이 서로 연결되어 대역폭을 경쟁하게 되는 상황을 현재의 대역폭 상황에 능동적으로 적응해 보다 효율적인 대역폭 활용을 가능하게 하는 USB 통신대역폭 사용 모델을 제안하였다. 제안된 모델의 시뮬레이션 결과로 대역폭 경쟁의 상황보다 대역폭 현재 상황에 적응해 가면서 대역폭을 사용하는 것이 대역폭의 전체 데이터 수신량이나 수신률에 있어서 모두 보다 높은 결과치를 보여주고 있다.

USB는 현재에도 가장 인기 있고 사용하기 편리한 사용자 인터페이스로 널리 사용되고 있고 USB의 활용도 보다 넓고 보편화되어 다양한 형태로 적용되리라 판단된다. 이와 더불어 USB의 활용이 디바이스 간 다중 인터페이스와 멀티미디어 동영상 전송과 같은 대용량의 정보 전달이 필요로 하는 분야에서의 활용은 무엇보다도 전송대역폭의 효율적 활용

이 우선적 과제가 될 것이고 이에 본 논문에서 제안된 보다 적응성 있는 모델을 적용한다면 USB를 이용한 다중 멀티미디어 환경에서의 디바이스 간 통신 인터페이스 성능향상이 기대된다.

보다 정확한 성능향상을 위해 다수의 USB 디바이스들의 연결 시 각 디바이스 특성상의 전송지연이나 등시성(Isochronous)전송에서의 전송 에러에 의한 전송지연회수의 임계값(Threshold) 적용에 최적의 임계값 선택 알고리즘을 개발하여 이 또한 적응성을 가지는 것이 필요하다.

▌참고문헌▐

[1] Wooi Ming Tan, Developing USB PC Peripherals Second Edition Using the Intel 8x930Ax USB Microcontroller, Annabooks, February, 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] 전세일, 이두복, “USB 인터페이스를 이용한 데이터전송 프로그램 개발”, 한국정보처리학회 논문지, 제7권 제5호, pp.1553-1558, 2000.
[6] 김형훈, USB GUIDE Universal Serial Bus, Ohm, 2002.