International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 9, November- 2012

Comparision of Data Transfer Protocols over USB

Megha Dey

Abstract

In the modern day, data transfer between a PC and a mobile storage device over the USB interface represents a very common user operation. The popularity of USB compliant devices is an indication of the modern user's need for a fast, large capacity and easily accessible system for data storage. Until recently, the USB Mass Storage Class(MSC) was the underlying class protocol being used, but with many of the intelligent storage devices shifting to media transfer protocol (MTP), it is important that we study these protocols, evaluate their main features and behaviour through various metrics including architecture, implementation, data transfer initialization, robustness, throughput, user friendliness and market penetration.

1. Introduction

USB provides a standard, low cost option along with an expandable hot pluggable mechanism for peripheral devices. USB data transfers takes place between a host (PC) and device (storage medium).Transfers takes place every time data moves between the USB host controller and a buffer on the device known as endpoint. This happens via logical communication channels known as pipes. Data transfers take place through 'Bulk' endpoints, which can send large amount of data (max 512 bytes per transaction).A bulk transfer consists of one or more IN (device to host) and OUT (host to device) transactions. As USB is a host-centric protocol, all transactions are initiated by the host.

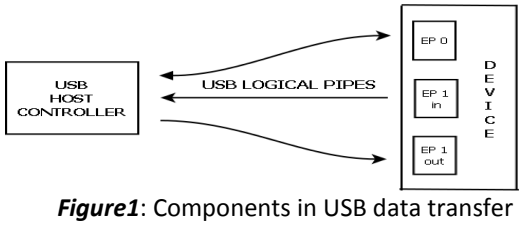


Figure1: Components in USB data transfer

USB MASS STORAGE CLASS (MSC) PROTOCOL

The MSC protocol was introduced in 1999 by the USB implementers forum (IF).It first came in the form of CBI (control/bulk/interrupt) protocol, but was soon taken over by the BOT (Bulk Only Transfer) or BBB (bulk/bulk/bulk) protocol. It is used by almost all flash based devices like thumb drives, hard disks and until now in mobile devices. This protocol provides a recognizable interface to the storage device embedding SCSI commands inside class structures CBW (Command Block Wrapper) and CSW (Command Status Wrapper).SCSI is a standard used by hard disks etc. to define a I/O bus to interconnect computers and peripherals. Hence, USB MSC devices appear as external disks to the host.

MEDIA TRANSFER PROTOCOL (MTP)

MTP was initially introduced by Microsoft in 2007 and was adopted by USB IF in 2011. This protocol enables data exchanges between initiators (PC) and responders (intelligent storage devices like MP3 players, tablets and Smartphones), also enabling synchronization of device content. MTP makes an effort to develop secure, feature rich user interface on handheld devices. Unlike MSC, storage devices appear as a portable device instead of a drive. "Media" in Media Transfer Protocol does not just mean audio/video; it encompasses all binary data including text.

2. Comparative Analysis

2.1. Architecture

USB implements a layered architecture. Layers 1 to 3 are common for BOT and MTP.

LAYER 1 represents the USB hardware controllers.

LAYER 2 represents the USB Controller Driver which is hardware specific. It is responsible for initializations, register accesses and interrupt handling of the chosen controller hardware.

LAYER 3: contains the generic USB functionality and is responsible for the detection and enumeration of connected devices. It also routes the packets to a particular class driver. It is independent of the underlying hardware controller used.

In MSC BOT:

Layer 4 implements Mass Storage BOT Protocol and SCSI Command handling.

Layer 5 consists of the file system driver on the host and the user application on the device side.

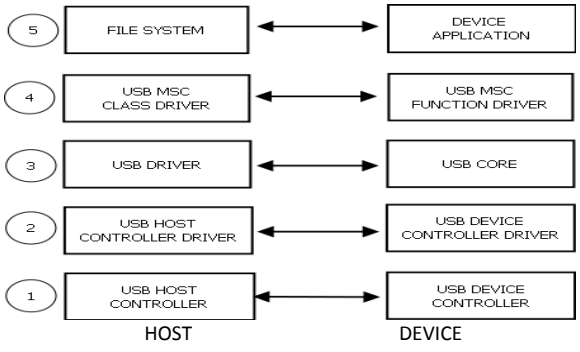


Figure 2: USB MSC BOT Architecture

In MTP:

Layer 4 is responsible for MTP class specific requests, MTP command decoding, and responding.

Layer 5 has the user application on the host side and the file system on the device side.

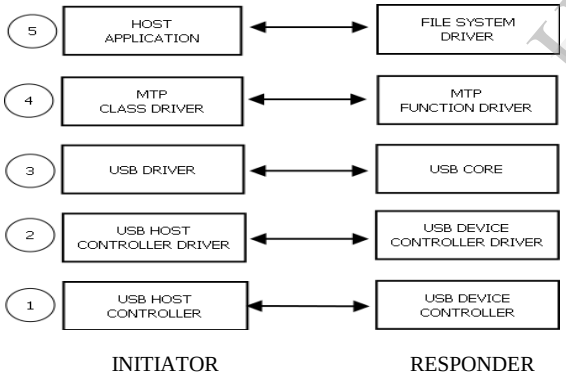


***Figure 3*: MTP Architecture**

2.2. Device Implementation

MSC BOT

The commands coming from the USB host controller (CBW) are unwrapped by the USB MSC protocol decoder and SCSI decoder and transmitted to the storage media . The storage response is then converted into SCSI commands and then wrapped in a USB specific format,and sent to the USB host controller.

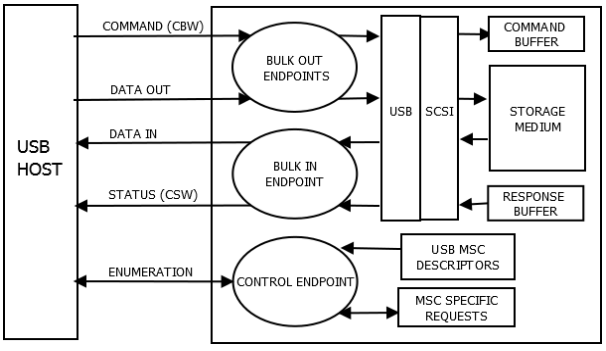


Figure 4: USB MSC BOT Implementation

Each data transfer(read/write) follows a Command-data-status pattern.A mass storage device contains:

- A bi-directional control endpoint for enumeration.
- A bulk OUT endpoint, to transfer wrapped SCSI commands (CBW) and data from host to device.
- A bulk IN endpoint, to send wrapped SCSI status through CSW and data from device to host.

MTP

In addition to 1 control and 2 bulk endpoints, MTP makes use of an interrupt endpoint to report device initiated events.

The MTP commands, data and response are wrapped in a specific container format. All Device content (songs, images, videos) are represented by objects and object handles are used in order to reference a logical object on the device.

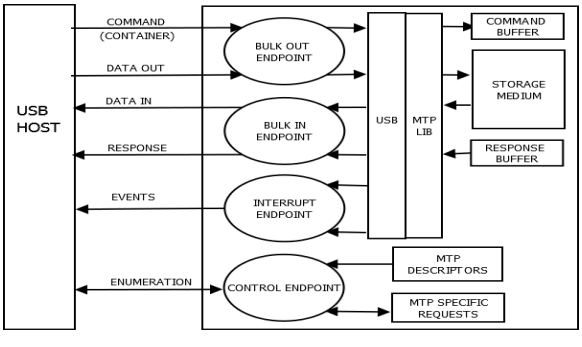


***Figure 5*: MTP Implementation**

Thus, MTP enables generic object transfer between devices.

2.3. File System Handling

From both the architectures, we see that in the case of MSC, the file system handling is a function of the host. Once a MSC device is connected to the host, it appears as any other block device and the host has unrestricted access to it. For each file transfer, the host must decode the FAT (File Allocation Table) to know the locations from where read or write should happen.

On the other hand, in case of MTP, the file system accesses are totally managed by the device CPU. The processor manages the interpretation of the MTP commands through the software stack as well as the accessing of data from the attached storage device. Hence with MSC, content storage is managed by the PC and with MTP, the device CPU actively manages content storage.

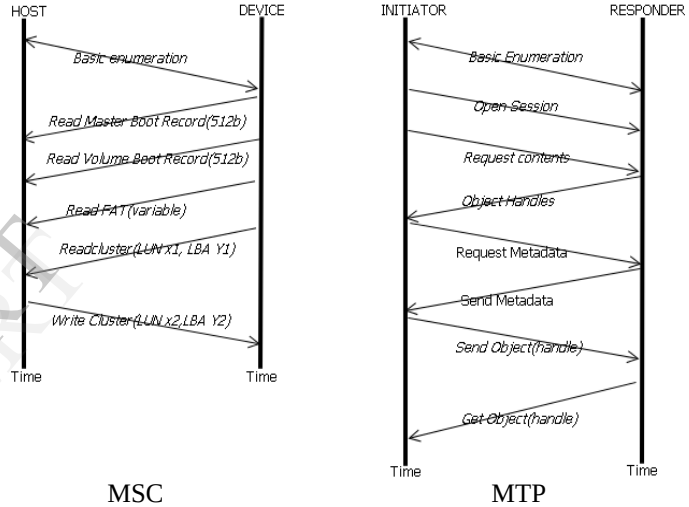
MSC gives a very low-level interface to the host, allowing only a single entity (host or device) to communicate with the storage at any given time, while in MTP the host and the device can have simultaneous access to the device, sharing the file system. Thus, besides enabling data transfer, MTP intrinsically enables some command and control of the connected device.

2.4. Data Transfer Initialization

Once the device is connected, during enumeration, USB descriptors are requested by the host, to determine the capabilities and requirement of the device.

In case of MTP, on accessing the device after enumeration, a new session is initiated and all the associated metadata such as file creation time, modified time, file size, folder size etc are passed on to the PC; thereby the Initiator builds model of responder contents (metadata based enumeration). Since all content are represented as objects, all the object handles are also passed onto the host. All this data must be stored permanently on the device. An integrated database is used for this and it should be optimized for easy retrieval. Storing all object metadata in memory can consume prohibitive amounts of device memory, possibly destabilizing the device. The amount of metadata depends on the number and the type of files. This organization of objects with properties enables easy enumeration of the objects, without having to interpret the underlying file system. Once all the metadata is shared to the host, object exchange starts.

In the case of MSC, there is no metadata transfer involved. After enumeration, information about the Master Boot Record (where the host learns about the whole memory and the partitions) and Volume Boot Record (where host gets the entry point to the FAT for each partition) are sent from the device. Then to read/write data, the respective LBA (logical block address is sent to the host or device.



MSC MTP

Figure 6: Data transfer initialization

Thus, MTP has a more elaborate data transfer initialization sequence compared to MSC.

2.5. Throughputs

Probably the most important parameter for any data transfer protocol is the measurable throughput. Though USB 2.0 has a maximum theoretical speed of 480Mbps, only a fraction of this value is achieved. USB data throughput depends on various factors:

On Host:

- Register and memory Cycle timing
- Maximum payload (512 for USB 2.0 bulk)
- CPU and USB host controller load

On Device:

- Buffer size of the device
- Type of buffering(single/double)
- Storage media type (internal/external)
- NAK'S on the bus

Other factors:

- Data transport protocol (MSC/MTP)
- Interrupt latency
- File system
- Operating system

USB protocol overheads (Refer [1]):

- Start and End of Frames (SOF's and EOF's)
- Fixed bandwidth for control transfers (20%)
- Bit stuffing

MSC follows a simple command, data and status state machine. For every 64 KB data transfer in windows and 120 KB transfer in Linux, extra 44(31+13) bytes are also sent (CBW+CSW), along with an interrupt to notify the upper layers that some data has been read/written. Also, there is a delay between each of the 3 phases (around 500 µs).

MTP also uses the command data and status pattern, but the object size is not limited; it can extend even upto 4GB.Thus, the actual number of bytes which are sent or received by the device will be relatively higher than the actual file size in the case of MSC. Also, the number of interrupts serviced in MSC will also be significantly higher than MTP.

Thus, at the USB protocol level, it is expected that MTP throughput will be higher than MSC. To validate this assumption, the following experiment was performed.

Experimental Setup

This experiment focuses on data transfer between a PC and a mobile platform, using high speed USB (USB 2.0). On the PC side, both Windows7 and Ubuntu11 operating systems have been used. The mobile platform used is ST Ericsson's "Snowball" development board version of android and Samsung Galaxy nexus tablet, both running the Jellybean flavour of Android. While the Snowball platform supports both MTP and MSC, the Nexus supports only MTP. The snowball board has 8GB internal memory and a SD card slot; the Nexus has also has 8 GB internal storage with no SD card slot.

All the values reported are based on transfers involving only the internal storage. All the USB transactions were captured using an Ellisys Explorer 2.0. The time taken for the transfer was measured from the first 512 bytes transferred to the last 512 bytes captured [OUT transaction during write and IN transaction during read]. In Ubuntu, the MTP devices were mounted as a directory using [7].Throughputs were calculated for single file transfers as well as multiple file transfers for 2 file sizes (300MB, 1GB) and then averaged out to obtain the final throughput.

The read (transfer from device to host) and write (transfer from host to device) throughputs are as indicated below in Mbps:

Table 1: SNOWBALL BOARD (MSC) THROUGHPUTS:

FILE SIZE	WINDOWS 7		UBUNTU 11	
	READ	WRITE	READ	WRITE
300MB	165.3	96.3	87.43	135.42
1 GB	158.2	104.2	105	85.24

Table 2: SNOWBALL BOARD (MTP) THROUGHPUTS:

FILE SIZE	WINDOWS 7		UBUNTU 11	
	READ	WRITE	READ	WRITE
300MB	166.7	80.1	151.7	70.4
1 GB	161.9	71.93	145.1	62.3

Table 3: SAMSUNG GALAXY NEXUS (MTP) THROUGHPUTS:

FILE SIZE	WINDOWS 7		UBUNTU 11	
	READ	WRITE	READ	WRITE

300MB	137.91	68.71	91.535	69.13
1 GB	144.49	59.88	103.265	55.21

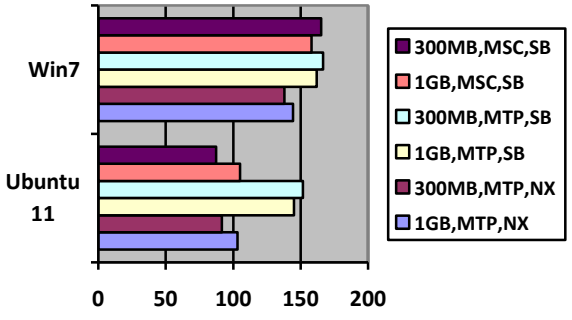


Figure 7: Read throughputs

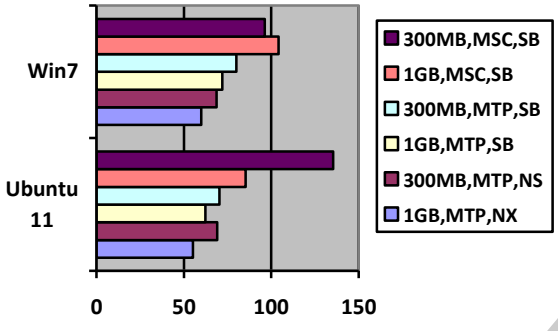


Figure 8: Write throughputs

SB:Snowball

NX:Nexus

Throughput Results:

Though it is theoretically expected that MTP must giver higher throughputs compared to MSC, it is seen that this is not always true. The key insights drawn from this experiment are as follows:

1. MSC has higher write speeds than MTP on both the platforms.
2. Although the read speeds are comparable on windows7, MTP has higher read speeds on Ubuntu than MSC.
3. Even though double buffering is enabled in Nexus, both the read and write speeds are lesser than Snowball:From the bus transactions, it is seen that there is a lot of data corruption due to CRC16 errors. This results in the re-transmission of data every time data corruption occurs.

2.6. Robustness

Any protocol is primarily measured by its robustness, i.e. ability to recover or handle error conditions. In this respect, MSC is less efficient compared to MTP. Since MSC involves mounting and unmounting each time the device is plugged in, there are chances of data corruption if device is plugged out; especially when the performance mode is used (i.e. caching is enabled. Since MTP allows exclusive read/write access from the computer, there are fewer chances of data loss and corruption for those who do not unmount before removing the USB cable. When a FAT file system is attached to a host computer, the file system is "owned" by the host and thus the device itself cannot for example add, remove, play or rename files during this time without risking corruption of the file system. MTP does not necessitate such device lock-down.

2.7. User Friendliness

When a mobile device is in mass storage mode, you cannot access the storage using the device until you unmount it from the PC. With MTP, a handheld retains access to storage even while plugged into a PC; users can still make a phone call or play music and videos while connected, instead of facing a frozen screen.

Also, unlike MSC, MTP enables monitoring of device initiated events and changes in device properties. This is done via the interrupt endpoint. Hence, the user can take a picture while there is any data transfer going on and the metadata associated with the image is directly updated to the host.

Additionally, MTP Initiators have the ability to identify the specific capabilities of device with respect to file formats and functionality.

Lastly, MTP is also more secure than MSC, as it allows digital rights management (DRM) protected content to be transferred to a host.

Thus, MTP is more user-friendly than MSC.

2.8. Market Penetration

Since MSC BOT was introduced much earlier than MTP, the number of mobile devices using MSC

currently clearly outnumbers MTP. But, with various mobile players like Android switching to MTP, there is likely to be a boom in MTP devices in the near future. Also with MTP, for devices with lots of internal memory, a manufacturer no longer needs to come up with partitions between the USB storage and internal storage. The storage is unified and can be used for either applications or media, depending on what the user wants to put on it. But there are some hurdles which MTP needs to address. MSC drivers are by default present in all flavours of Windows and Linux. While MTP drivers are integrated in windows 7 onwards, one needs to have windows media player 10 or higher on windows XP. On Linux, one needs to install external libraries like Libmtp etc. and 3rd party applications to access MTP. MTP was initially designed for stand-alone, hard-drive based media players. Unlike mobile devices, stand-alone players don't typically have removable media, vulnerability to malicious code or have to integrate seamlessly with other tasks the user performs. As long as manufacturers can overcome these challenges and optimize MTP for large storages and multifunction devices, more and more mobile devices will switch to MTP.

3. Conclusion and Summary

The results of the comparative study between USB MSC and MTP can be summarized using the following table (**Table 4**).

USB MSC and MTP represent two classes for efficient data exchanges. While MSC has been the de-facto standard being followed until the recent past, it fails to address the needs posed by modern day intelligent storage devices. In this study, we have found that even though we expect MTP to have higher throughput, it is not so as the MSC ecosystem is more stable and mature. But MTP scores when it comes to robustness and user friendliness. Though flash based devices will continue to use MSC, it is expected that the freedom of data management and the safe transfer of data will enable more and more devices supporting MTP.

Table 4:Comparative Analysis Results

Property	***MSC***	***MTP***
Architecture	Layered	Layered
Implementation	1 Control, 2 Bulk Endpoints	1 control, 2 Bulk and 1 Interrupt endpoint
File System Handling	Host	Device
Data Transfer Initialization	Less elaborate	More elaborate
Robustness	Less	More
Throughput (theoritical)	Less	More
Throughput (practical)	More	Less
User Friendliness	Less	More
Market Penetration	More	Less

4. References

[1] *USB 2.0 Specification, USB-IF 2005*
[2] *Mass Storage Bulk Only 1.0 spec 1999*
[3] *Media Transfer Protocol v.1.1 Spec 2011*
[4] *Steve Kolokowsky and Trevor Davis, "Introduction to MTP:Media Transfer Protocol",Cypress Semi-Conductors*
[5] *Abhishek Bit, Dr. Martin Orehek, Waqar Zia, "Comparative Analysis of Bluetooth 3.0 with UWB and Certified Wireless-USB Protocols", Proceedings of 2010 IEEE International Conference on Ultra-Wideband (ICUWB2010)*
[6] *Blake Manders, Donby Mathieu "Presentation on Media Transfer Protocol Implementation Details",Microsoft Corporation*
[7] *"http://jnm-tech.blogspot.in/2012/05/how-to-access-archos-g9-on-ubuntu.html"*
[8] *Mark McLemore, Harold Drews "MTP Responder Development guide",Microsoft Corporation*
[9] *Steve Kolokowsky and Trevor Davis "Mass Storage Class vs. Media Transfer Protocol", Cypress Semi-Conductors*
[10] *"USB Mass Storage Device Implementation.pdf", Atmel Corporation*
[11] *Jan Axelson,"USB Mass Storage:Designing and programming Devices and Embedded hosts.*