A method and apparatus of performing a data transmission from an electronic device or a peripheral device of an electronic device to a peripheral device of a remote electronic device is disclosed. One example method of performing the data transmission may include transmitting data designated for the remote peripheral device to a local virtual device object. The data that is received by the local virtual device object is transmitted via at least one communication interface or peripheral device of the electronic device to at least one remote communication interface or peripheral device of the remote electronic device. The data arriving at the least one remote communication interface or peripheral device of the remote electronic device is received by a remote virtual device object and transmitted to the designated remote peripheral device.

18 Claims, 5 Drawing Sheets
References Cited

OTHER PUBLICATIONS

* cited by examiner
FIG. 1

SYSTEM MEMORY
- ROM 131
- BICS 132
- RAM 133
- OPERATING SYSTEM 134
- APPLICATION PROGRAMS 135
- OTHER PROGRAM MODULES 136
- PROGRAM DATA 137

PROCESSING UNIT
- VIDEO ADAPTER
- IEEE 1394 INTERFACE

SYSTEM BUS 106
- HARD DISK INTERFACE
- MAGNETIC DISK DRIVE INTERFACE
- OPTICAL DRIVE INTERFACE
- SERIAL PORT INTERFACE

MONITOR 100
- DEVICE 104

LOCAL AREA NETWORK 114
- WIDE AREA NETWORK 119

KEYBOARD 115
- MODEM 118
- REMOTE COMPUTER 120

APPLICATION PROGRAMS 123
- MEMORY 122
BEGIN

SEND DATA

RECEIVE DATA

QUERY TARGET DEVICE STATUS

NO

OK?

YES

TRANSMIT DATA

END

FIG. 4
BEGIN

501 RECEIVE DATA

502 BUFFER DATA

503 TARGET DEVICE?

504 SERIAL DEVICE

505 IEEE 1394 BUS NODE

506 TRANSMIT DATA

FIG. 5

END
INTERCONNECTION OF PERIPHERAL DEVICES ON DIFFERENT ELECTRONIC DEVICES

TECHNICAL FIELD OF THE INVENTION

The present invention relates generally to the use of peripheral devices and communication interfaces to communicate between electronic devices such as computers and, in particular, to bridging of peripheral devices via transport media that operates at various speeds and communication standards such as synchronous and asynchronous.

BACKGROUND OF THE INVENTION

A computer system is comprised of different components or devices that operate together to form the resultant system. Typical computer devices such as the central processing unit usually are supplied with the computer system initially whereas other devices can be installed into the computer system after the initial configuration of the system. The devices of the computer system are generally coupled together via interconnects which may be of several types, such as system and peripheral bus.

Networks are typically comprised of a variety of different electronic components or devices that are able to identify and communicate with each other by using an addressing and communication scheme that is known by each participant. The network allows computers to communicate with each other and share resources and information. Computer networks are made up of basic hardware building blocks to interconnect network nodes, such as Network Interface Cards, Bridges, Hubs, Switches, and Routers. A network card, network adapter or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a computer network. It provides physical access to a networking medium.

Device drivers are well known in the art. When a device is installed onto a computer by the user, a device driver has to be loaded to enable communication with the device. A device driver is software that is used to describe and control the device for the operating system. A virtual device driver is a specific type of device driver that, like a device driver for a physical existent device, has full access to the operating system kernel and is able to communicate directly to a physical port. In difference to device drivers for existent hardware a virtual device driver is loaded without a hardware device being detected by the system. A virtual device driver manipulates kernel mode code using existing hardware resources to emulate a device that is not present on the computer. A virtual driver is given more access than a traditional device driver because it is not restricted to talking to just one particular device.

Virtual device drivers are designed to handle hardware device contention between multiple processes and to translate or buffer data transfers from a virtual machine to hardware devices. A virtual machine is self-contained operating environment that behaves as if it were a separate computer. When two or more processes attempt to access the same device, some method of contention management must be used. A virtual device driver allows each process to act as though it has exclusive access to the device. For example, a virtual printer driver would provide the printing process with a virtual printer port, and characters written to the port would be written to a print spooler. The virtual device driver would then send the job to the printer when it becomes available. Another method would be to assign the physical device to only one process at a time, so that when a process attempts to access the device while it is in use, the virtual device driver does not pass the request to the actual hardware, and the process operates as though the hardware did not exist. Virtual device drivers also virtualize input/output to the device and for example a virtual network device driver translates this information into commands to be sent across a network to a hardware server.

A technique that allows error free communication between devices that obligatory require exchange of protocol status information's faster than provided by the underlying network transport is not currently known in the art. However, such a technique would offer significant advantages over the prior art. A method for bridging multiple network segments and exposing the multiple network segments as a single network to a higher level networking software on a bridging computing device as described in U.S. Pat. No. 7,089,335 is known in the art. However, if this method is used to connect devices that work with a time critical protocol and the transmission speed of the network segments fall under a certain value communication errors occur at the devices.

As disclosed in U.S. Pat. No. 6,968,307 a technique for creation and use of virtual device drivers on a serial bus is already known. Beside the creation of virtual device drivers also an extension of the peripheral bus even over LAN and WAN networks is described. The method emulates the peripheral bus itself with all its typical features and characteristics. This method does not give an acceptable answer to the question how devices can communicate successful if the underlying network media itself does not meet the requirements of the peripheral bus in concern of speed and timing.

An extension of fast peripheral buses like the IEEE 1394 “Firewire” bus over comparatively slow transport media such as wireless networks as disclosed by the IEEE 1394 Trade Association in December 2001 allows error free communication between endpoints of such an extended peripheral bus in terms of bus availability and clock time synchronization. However, a solution for devices that obligatory require the exchange of protocol information’s at a speed that is higher than the wireless network is not included.

The IEEE 1394 Trade associations white paper “Networking IEEE 1394 Clusters via UWB over CoaxialCable” discloses and covers many parts of communication between devices on a extended peripheral bus inclusive all timing requirements of video and audio devices. However, this disclosure is solely based on using an underlying network technology that is quite as fast as the peripheral bus itself. Slow, dropped out or unavailable networks are not covered by this disclosure.

Virtualization technologies from various manufacturers already allow the creation of additional virtual hardware resources such as network interface card, hard disk and computer system (Guest machine) on a physical existent computer system (Host machine). This technique emulates a machine with it’s physical properties. By establishing interconnections between the physical devices of the host and the devices of the emulated guest devices of the guest can be attached to similar physical devices of the host. However, devices on peripheral buses of the host machine such as nodes of a serial bus cannot be attached to nodes of the peripheral bus of the guest machine because the peripheral bus of the host is independent of the guest’s bus. Even if possible, if the architecture of the guest’s operating system fundamentally differs from the host operating system the attachment of newly added devices on peripheral buses of the host to the guest’s peripheral buses most likely fails because the device is announced by the operating system and not the machine.
In Digi International’s AnywhereUSB Remote I/O Concentrator documentation another method is disclosed that is applicable for accessing physically existent USB devices on an external hardware device which is attached to a network cable from a remote computer that is also connected to a network. At the remote computer a service is used that connects to the external hardware by using the TCP/IP network transport protocol of the remote computer. Thru the network connection the service establishes a socket connection to the device. A hardware description information of each external USB connector is installed on the local computer. The local serial bus of the computer is extended with these new nodes and makes them part of the local serial bus. By design, this technique extends the local serial bus of the computer to the external device by emulating a wired connection over the network. The external device itself gets a part of the local serial bus. This approach requires that the external device with its USB connectors can be reached by the network and provides information’s about the installed connectors to the local service. If USB connectors are part of another computer’s local serial bus or different types of connectors are used this method fails because the computer does not provide information’s about the nodes installed on his local serial bus thru the network.

SUMMARY OF THE INVENTION

A peripheral device connected to a local electronic device which is connected to at least one communication network can communicate with a peripheral device attached to a remote electronic device as if the remote peripheral device was locally attached. Data designated for the remote peripheral device is received by a local virtual device object and transmitted to the remote electronic device via at least one of the electronic devices communication interfaces or peripheral devices. Data received by the remote electronic device’s communication interface or peripheral device is written to the peripheral device at the remote electronic device by a virtual device object. For compensation of different transfer speeds or outages between the peripheral device and the communication interface or another peripheral device the virtual device provides the ability to utilize the virtual devices emulation driver that is attached to the virtual device object as an I/O buffer. As the invention provides a generic method for virtualization of a remote peripheral device it works independent from specific types of devices and Operating Systems. As the invention works with synchronous and asynchronous communication standards and does not require a specific network transport protocol it can be used with any available communication interface or peripheral device of the electronic device.

One embodiment of the present invention may include a method of data transmission from an electronic device or a peripheral device of an electronic device to a peripheral device of a remote electronic device. The method may include transmitting data designated for the remote peripheral device to a local virtual device object, the data received by the local virtual device object is transmitted via at least one communication interface or peripheral device of the remote electronic device. The data may also provide that the data received at the least one remote communication interface or peripheral device of the remote electronic device is received by a remote virtual device object and transmitted to the designated remote peripheral device.

A peripheral device configured transmit data to a peripheral device of a remote electronic device. The apparatus may include a transmitter configured to transmit data designated for the remote peripheral device to a virtual device of an electronic device. The apparatus may include a receiver configured to receive data designated for the remote peripheral device and transmit the data to a peripheral device of a remote electronic device. The apparatus may also provide that the data received at the least one remote communication interface or peripheral device of the remote electronic device is received by a remote virtual device object and transmitted to the designated remote peripheral device.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 is a block diagram of an exemplary operating environment.

FIG. 2 is a block diagram of a system for transmission of data between electronic devices.

FIG. 3 is a flow chart illustrating a method of creating a virtual device in accordance with the present invention.

FIG. 4 is a flow chart illustrating a method of transmission of data from a peripheral device in accordance with the present invention.

FIG. 5 is a flow chart illustrating a method of reception of data by a device in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

It will be readily understood that the components of the present invention, as generally described and illustrated in the figures herein, may be arranged and designed in a wide variety of different configurations. Thus, the following detailed description of the embodiments of a method, apparatus, and system, as represented in the attached figures, is not intended to limit the scope of the invention as claimed, but is merely representative of selected embodiments of the invention.

The features, structures, or characteristics of the invention described throughout this specification may be combined in any suitable manner in one or more embodiments. For example, the usage of the phrases “example embodiments”, “some embodiments”, or other similar language, throughout this specification refers to the fact that a particular feature, structure, or characteristic described in connection with the embodiment may be included in at least one embodiment of the present invention. Thus, appearances of the phrases “example embodiments”, “in some embodiments”, “in other embodiments”, or other similar language, throughout this specification do not necessarily all refer to the same group of embodiments, and the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

Example embodiments of the present invention may be more vividly described with reference to FIGS. 1-5. FIG. 1 is a schematic diagram of a conventional digital electronic device that can be used in association with various embodiments of the present invention. The electronic device includes a processing unit 101, a system memory 130 and a system bus 105 that couples various system components including the system memory to the processing unit. System bus may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. System memory includes a read only memory (ROM) 131 and a random access memory (RAM) 133. A basic input/output system (BIOS) 132 containing the basic routines that help to transfer information between ele-
ments within the electronic device 100, such as during start-up, is stored in ROM 131. The electronic device 100 also includes a hard disk drive 107 for reading from and writing to a hard disk, an optical disk drive 111 of an optical disk interface 110 for reading from or writing to a removable optical disk 116, such as a CD ROM or other optical media and a magnetic disk drive 109 for reading from or writing to a removable magnetic disk 115. Magnetic disk drive 109, Hard disk drive 107 and optical disk drive 111 are respectively connected to the system bus 105 by a magnetic disk drive interface 108, a hard disk drive interface 106 and an optical disk drive interface 111. The drives and their associated media provide nonvolatile storage of electronic devices readable instructions, data structures, program modules and other data for the electronic device 100.

Of course other types of media readable by electronic devices which are storeable that is accessible by an electronic device such as flash memory cards, digital video disks, magnetic cassettes, random access memories (RAMs), read only memories (ROMs), and the like, may also be used in the exemplary environment.

A number of program modules can be stored on the hard disk, magnetic disk 115, optical disk 116, ROM 131 or RAM 133, including an operating system 134, one or more application programs 135 and 123, other program modules 136, and program data 137.

A user can enter commands and information into the electronic device 100 through input or selection devices, such as a keyboard 119 and a pointing device 117. The pointing device 117 may comprise a mouse, touch pad, touch screen, voice control and activation or other similar devices.

These and other input devices are often connected to the processing unit 101 through a serial port interface 112 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port or a universal serial bus (USB).

A monitor 104 or other type of display device is also connected to system bus 105 via an interface, such as a video adapter 102. In addition to the monitor, electronic devices typically include other peripheral output devices (not shown), such as speakers and microphones.

An additional serial port in the form of an IEEE 1394 interface 103 may also be provided. The IEEE 1394 interface 103 couples an IEEE 1394 compliant serial bus 138 to the system bus 105 or similar communication bus. The IEEE 1394 compliant serial bus 138, as known in the art, allows multiple devices 139 to communicate with the electronic device 100 and each other using high-speed serial channels.

The electronic device 100 can operate in a networked environment using logical connections to one or more remote electronic devices, such as a remote electronic device 121. A remote electronic device 121 typically includes at least some of the elements described above relative to the electronic device 100, although only a memory storage device 122 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 114 and a wide area network (WAN) 120. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the electronic device 100 is connected to a local network 114 through a network interface or adapter 113. When used in a WAN networking environment, a local electronic device 100 and remote electronic device 121 may both include a modem 118 or other means for establishing a communications over wide area network 120, such as the Internet. Modem 118, which may be internal or external, is connected to system bus 105 via serial port interface 112.

In a networked environment, program modules depicted relative to the electronic device 100, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the electronic devices can be used. The existence of any of various well-known protocols, such as TCP/IP, "ETHERNET", FTP, HTTP and the like, is presumed, and the system can be operated in a client-server configuration to permit retrieval of web pages from a web-based server. For example, in one example embodiment of the present invention, the remote electronic device 121 is a server having stored thereon one or more files that may be accessed by the local electronic device 100. Procedures of the example embodiments of the present invention described below can operate within the environment of the electronic device shown in FIG. 1.

Although the example embodiments of the present invention are generally applicable to an electronic device operating in accordance with the description shown above, the invention is applicable to any electronic device that is able to uniquely identify attached peripheral devices either by a reserved memory area or a unique node id, such as, when using peripheral devices that are connected to a peripheral interface that uses direct memory access control methods. In FIG. 2, there is a system that maybe used in accordance with an example embodiment of the present invention. A peripheral device 200 may be connected to an electronic device 201. The electronic device 201 comprises a peripheral device interface 202, which manages communications between the physical bus 203 and the peripheral device 200. The peripheral device 200 also includes a processing unit 204 coupled to a peripheral device interface 208 connected to another peripheral device 209. The peripheral device 200 has a link to a virtual device object 205 which has a link 212 to a communication interface 207 and another link 211 to the peripheral device 209.

A user of the electronic device 201 has the option of creating a virtual device object 205 that, after completing the remaining operations represents an image of the remote peripheral device 215 plugged into the remote electronic device 214. At creation, the virtual device object loads an emulation driver 206 appropriate for a node of the peripheral bus of the selected local attached peripheral device 200. Similar to the electronic device 201, the remote device 214 includes a peripheral device 213 in communication with a peripheral device interface 216 which is connected to a processing unit 217, another peripheral device interface 221 of another peripheral device 222. A virtual device object 218 has a link 223 to the peripheral device and operated by an emulation driver 219. A communication interface 220 communicates via link 224 to the virtual device object 218, which has a link 225 to the peripheral device 222.

The user may want to emulate more than one device. In this case, the user would repeat the process by creating a second virtual device object 205 that, after completing the remaining operations represents an image of the remote peripheral device 215 plugged into the remote electronic device 214.
or the remote electronic device to be plugged in or connected to create a virtual device object. A user mode application sends a request that tells the electronic device to create a virtual device object with certain peripheral device properties. The virtual device object loads an emulation driver that supports the target functionality of a node of the selected peripheral interface or bus. If the remote electronic device with the remote peripheral device attached to it is connected to the local electronic device the virtual device object is already present and is immediately capable of representing the complete functionality of the peripheral device of the remote electronic device.

Conventionally, the electronic device would not be able to represent to its peripheral bus nodes functionality other than that of a physical device attached to the local peripheral bus.

In accordance with one example embodiment of the present invention, the device 200 is assumed to be a mass storage node attached to an IEEE 1394 bus. The electronic device 200 would have a device driver (USB mass storage device driver) that enables communication with the device 200. The user can create a virtual device object 205 that represents an additional member of the peripheral device class of the peripheral device 200 even though the device is not attached to the local electronic device.

At the local electronic device a virtual device object may be created by modifying the installation files of the peripheral device 200. When an IEEE 1394 bus node is detected, an additional virtual device entry is automatically created in the registry. The virtual device object 205 uses an emulation driver 206 for communication with the device 200.

In addition to being able to emulate multiple peripheral devices at one time and not requiring that a device be plugged in to emulate the device, another example embodiment of the present invention permits “native” communication between physical existent and virtual devices across a peripheral bus. In the previous example, the peripheral device 200 can communicate using “native” language because it believes it is communicating with a node of the local peripheral bus instead of a remote electronic device. In the present example embodiment of the present invention no translations are necessary because the peripheral device 200 communicates directly with the virtual device object 205.

In FIG. 3, a method of creating a virtual device is shown, according to an example embodiment of the present invention. At operation 300 a generic virtual device object is created by the operating system of the electronic device that may be initialized by the existing peripheral device itself that sends it’s peripheral device properties. After the querying device class is received 301 the peripheral devices attached to the peripheral interface or bus are enumerated 302. In case of a peripheral interface like the serial port interface enumeration is done by the operating system followed by the creation of a new primary node 303. In case of a node of a 1394 bus the driver enumerates the existent nodes by reading the nodes from the control status register and then adds a new node by assigning a node id and writing the appropriate information’s to the CSR. In such case the driver issues a reset 304 of the peripheral bus to allow the nodes an enumeration of the new device.

After setup of the primary node the second binding of the virtual device is set. This requires querying available target devices 305. Such an operation may either be done by the user or by using a predefined setup file that automatically selects from available communication interfaces 207 or peripheral devices that may be connected to a remote electronic device. After querying the available devices a device can be selected 306 from a list of available interfaces like for example a network interface 307 a serial port interface 308 or an IEEE 1394 interface 309 and be set 310 as the primary target device.

After the primary target device is selected additional secondary devices may be selected 311 to guarantee error free communication even if the primary device fails. This process can be repeated several times for each device the electronic device should use. In case of using the method to emulate a node of a serial bus a bus reset can be forced after the device object’s exposition. This bus reset causes all devices or nodes attached to the serial bus to enumerate each other. Any other node may now see the new node, the device chosen to emulate.

The method of creating a virtual device as shown in the example embodiment of FIG. 3 can also be initialized by a request in the form of a data structure that is sent to the application program interface (API). The request to generate an additional node can be sent by an upper level driver that is already loaded for an IEEE 1394 device. The request could also be sent by an application upon user request. Using a request to the 1394 bus driver software can pass a data structure that causes the bus driver itself to add a virtual device and to fill in the data structure required for the device. By setting an instance ID for the device the instance of the device can be identified.

For enumeration of the emulation driver a string may be passed to the driver that is used to generate the PnP ID’s. To allow a virtual device on an IEEE 1394 bus to remain present despite a subsequent hardware or software reboot it may be necessary to pass a flag to the driver that keeps the device’s state persistent. This will provide the virtual device object to be reported as existent after a reboot. The electronic devices operating system believes the emulated device is present. In other words, the electronic devices operating system can then “see” the emulated device.

Allowing the emulated device to be seen allows any other node on the bus to communicate “natively” with the device rather than using an additional node that performs as a server/translator between the devices. Generally, virtual device objects and the respective drivers have the same access to the driver of the peripheral bus as would a physical device object and its respective driver.

Because there is no physical target device there are differences in behavior with a virtual device object because there is no physical target device. Normally the peripheral bus driver fills in the target node identifier and the appropriate packet size and transfer rate using information from the enumeration procedure with a particular device. Due to the fact that there is no physical device the virtual device object has to provide all packet information because there is no target device node. For example, a read/write request will be intercepted and the virtual device object will fill in the address information for the request. The bus driver does not overwrite any fields.

A request for allocation of a range exhibits different behavior if addressed to a virtual device object. All address allocations from an emulation driver will implicitly have the flag to broadcast access enabled if post notification on the address range is required. This is done to allow any external node to access the address range used by the emulation driver to simulate the device.

In FIG. 4, according to an example embodiment of the present invention, a method of transmission of data from a peripheral device in accordance with the present invention is shown. Data 400 sent by the physical existent peripheral device to the virtual device object is received by the virtual device 401. As the virtual device object was generated based on descriptive information of the sending device the exchange of necessary protocol status information 402 may be provided
The virtual device object may create multiple bindings. The first operation is to enumerate the existent peripheral device classes on the electronic device. At this device, the primary node is the target for the desired data transfer. The primary node may be automatically assigned by the virtual device object or manually selected from a list of available peripheral devices from the user. Because there is no need for creation of additional new bus nodes or new virtual peripheral devices, a bus reset, after enumeration of the existent nodes on the target bus, is not necessary.

As described in FIG. 3, it is necessary to select at least one communication interface or peripheral device at the virtual device object to inform the virtual device object about possible interfaces for incoming data. When installing the virtual device object on the target electronic device it may be necessary to use the existent setup information of the sending device to guarantee that the unique node id transmitted by the sender provides incoming data to the desired peripheral device.

Since data may arrive from various interfaces the virtual device object's primary task is to analyze the incoming data for the node id specified in the virtual device object. As this may take some time, the virtual device object writes the incoming data to a buffer that may be built up, according to the cached bus nodes, which may be provided by the virtual device object in various ways.

The virtual device object may add its unique node id in the form of an identification string that may either be in plain text or, for binary data, in the form of a binary pattern. Due to the fact that this data is never directly received by the peripheral device at the remote electronic device this does not lead to any reception errors at the remote peripheral device.

If a communication interface in form of a network interface card is used, unique identification of the source data can also be provided by adding identification information, such as, a number or a single flag to the data exchange protocol used by the communication interface. When using a network interface as the communication media, a fixed communication port, or a range of ports can be defined that allow identification by the receiving device.

FIG. 5, shows a method of reception of data by a device in accordance with an example embodiment of the present invention. As previously described, data from a specific peripheral device may arrive at the receiving electronic device from multiple devices having either communication interfaces or peripheral connections, such as, cables or WiFi connections. To guarantee exact identification of the incoming data, the virtual device object at the receiving electronic device will analyze the incoming data.

The architecture of the virtual device object used at the receiving device is similar to the architecture of the sending device. As described in FIG. 3, the virtual device object may be based on a generic device object that is able to create a peripheral device or bus bindings by querying existent peripheral device classes of the device. If the addressed peripheral device at the receiving electronic device is a member of the same peripheral device class as the peripheral device at the sending electronic device, the virtual device object installed at the sending electronic device may be installed from a setup file generated by the initial device.

As shown in FIG. 3, the virtual device object may create multiple bindings. The first operation is to enumerate the existent peripheral device classes on the electronic device. At this device, the primary node is the target for the desired data transfer.

For most requests, the behavior is the same between virtual and physical devices. Although the invention has been described in relation to preferred embodiments, many variations, equivalents, modifications and other uses will become apparent to those skilled in the art. The scope of the present invention should not be limited to the specific disclosure but determined only by the appended claims.

What is claimed is:

1. A method of data transmission from an electronic device or a peripheral device of an electronic device to a remote peripheral device of a remote electronic device, the method comprising:
   - receiving a request at the electronic device to create a local virtual device object comprising properties of the remote peripheral device;
   - creating the local virtual device object by an operating system of the electronic device, the local virtual device object configured to operate on the electronic device and emulate the remote peripheral device of the remote electronic device;
   - transmitting data designated for the remote peripheral device to the local virtual device object, wherein the local virtual device object is configured to add source identification information to the data;
   - buffering the data at the local virtual device object;
   - transmitting a report message from the local virtual device object to the electronic device indicating the successful reception of data at the local virtual device object;
   - the virtual device object may attempt at least one other binding to the same device or another device until a successful binding is established.

2. A method of data transmission from an electronic device to a remote device, the method comprising:
   - receiving a request to create a virtual device object at the remote device;
   - establishing a communication link between the devices;
   - transmitting the virtual device object to the remote device;
   - receiving the virtual device object at the remote device;
   - the virtual device object may be consumed by the remote device in a manner similar to a physical peripheral device.

3. The method of claim 1, further comprising:
   - providing a unique identifier for the virtual device object;
   - transmitting the unique identifier with the virtual device object;
   - the virtual device object may be uniquely identified for purposes of communication and identification.

4. The method of claim 1, further comprising:
   - providing a capability for the virtual device object to communicate with other devices;
11. The apparatus of claim 9, wherein the designated remote peripheral device is identified by the payload data received.

12. The apparatus of claim 9, wherein at least one communication interface or peripheral device of the apparatus uses an asynchronous communication standard.

13. The apparatus of claim 9, wherein at least one communication interface or peripheral device of the apparatus uses a synchronous communication standard.

14. The apparatus of claim 8, wherein the peripheral device of the apparatus is connected to a peripheral bus.

15. The apparatus of claim 9, wherein a buffer size of the buffer dynamically increases or decreases according to a transfer speed of the peripheral device that transmits the data.

16. The apparatus of claim 15, wherein the buffer size dynamically increases or decreases according to the transfer speed of the communication interface or the peripheral device used for data transmission by the apparatus.

17. A non-transitory computer readable storage medium configured to store a computer program that when executed by a processor performs data transmission from an electronic device or a peripheral device of an electronic device to a remote peripheral device of a remote electronic device, the processor being further configured to perform:

receiving a request at the electronic device to create a local virtual device object comprising properties of the remote peripheral device;

creating the local virtual device object by an operating system of the electronic device, the local virtual device object configured to operate on the electronic device and emulate the remote peripheral device of the remote electronic device;

transmitting the data designated for the remote peripheral device to the designated remote peripheral interface and the remote peripheral device of the remote electronic device;

wherein the designated remote peripheral device is identified by a content of a communication protocol of the communication interface.

18. The apparatus of claim 9, wherein at least one communication interface or peripheral device of the apparatus uses an asynchronous communication standard.

19. The apparatus of claim 9, wherein at least one communication interface or peripheral device of the apparatus uses a synchronous communication standard.

20. The apparatus of claim 9, wherein the buffer size of the buffer dynamically increases or decreases according to a transfer speed of the peripheral device that transmits the data.
configured to identify and remove the source identification data of the received data; and
transmitting by the remote virtual device object a payload data of the received data to the designated remote peripheral device.

18. The non-transitory computer readable storage medium of claim 17, wherein a buffer size of the buffering dynamically increases or decreases according to the transfer speed of the communication interface or the peripheral device used for data transmission by the electronic device.

* * * * *