

BELCARRA USB RNDIS/CDC

1 INTRODUCTION

Many of today's and almost all of tomorrow's smart peripherals (PDAs, cell phones, pagers, GPS, etc.) have adopted the USB device port for communications with host systems. While many first generation units use USB for a minimal serial link (using proprietary protocols in many cases), the USB link is more and more being construed to be a tiny Ethernet segment, bringing higher speeds and greater flexibility. We suggest strongly that a smart peripheral, of whatever kind, should therefore adhere to popular protocols for Ethernet-like networking over USB.

1.1 *USB and Ethernet*

When initially considered, Ethernet over USB was aimed at the replacement of standard Ethernet cards or peripheral chips attached to the ISA (and later PCI) bus of the host system. In most host systems, especially today's desktop OS based systems, the addition of an actual Ethernet port connector via USB turns out to be one of the last uses of Ethernet over USB. Instead, the Ethernet protocol has been called upon to directly connect peripherals which don't have a hardware Ethernet port on them at all, to PC systems which already have a hardware Ethernet port in use for other things (typically LAN/WAN connection.) This "native" Ethernet is hardwired on the mother board or still on an add-on card – but is now shipped almost by default as the Internet and Ethernet have become the *de facto* standards for LAN connections in the time since USB was first proposed in the mid '90s.

The fact that Ethernet (and most often TCP/IP as well) is being used in this fashion brings complications to its use as the protocol for additional peripherals. As noted below, Belcarra, with our extended background in TCP/IP, has identified a number of areas where not only basic USB problems exist, but also TCP/IP problem areas; and we offer a number of unique solutions to these problem areas such as local DHCP service and 802.1 bridging.

1.2 *Two "Standards"*

There are two standards for Ethernet over USB, one *de facto*, and the other issued by the USB Forum.

The *de facto* and preferred standard, because it is officially supported by Microsoft is Remote NDIS (RNDIS). Any modern smart peripheral should offer this protocol to host PC's whenever possible.

The *de jure* standard, issued by the USB Forum, is known as CDC Ethernet.

Each of these standards imposes requirements on the UDC¹. A particular UDC may meet the requirements of:

- both RNDIS and CDC Ethernet
- RNDIS only
- CDC Ethernet only
- neither RNDIS nor CDC Ethernet

For more limited UDCs a variant of CDC Ethernet known as SAFE can be used.

This white paper introduces Belcarra Technologies' RNDIS/CDC software. This software package is an add-on to the Linux USB Device Framework². This product provides a USB device-side implementation (specifically a *function module*³) of the full range of these protocols:

- RNDIS
- CDC Ethernet
- SAFE.

Belcarra also publishes a host-side GPL CDC/SAFE solution (class/client module) for Linux and has a host-side CDC/SAFE driver in preparation for Mac OS/X.

Belcarra's RNDIS/CDC function module provides a solution that can support both RNDIS and CDC hosts without runtime reconfiguration. In combination with our forthcoming OS/X CDC/SAFE driver, we can provide a solution for all common host desktop operating systems using a single target runtime configuration.

2 WHAT IS RNDIS?

Remote NDIS⁴, generally known as RNDIS, is Microsoft's vendor-specific protocol for Ethernet-like devices. Microsoft supplies RNDIS driver stacks for all of its operating

¹ USB Device Controller, the portion of a SOC (System on Chip), which implements the USB Device port.

² This is the name that Belcarra is using to describe the existing GPL Linux device-side code originally issued by HP under GPL, and currently published by Sharp and Belcarra. The Belcarra version has been updated and we refer to it as Version 2. Version 1 (as published by Sharp) cannot support RNDIS.

³ Refer to our forthcoming Linux USB Device Framework backgrounder for more information on the terminology. For the present, the following definitions may suffice. A *class driver* or *class module* is a high-level module/driver on a host PC which implements support for a class of device (such as a mouse or keyboard – Human Interface Device aka HID). A *client driver* is a (high-level) host-side driver that implements support for a specific USB device. On the peripheral side, a function module or function driver matches a client or class module.

⁴ Network Driver Interface Specification – a Microsoft abstraction layer for networking

systems from Windows 98 to date. Windows Millennium was the first version of MS Windows to ship with RNDIS drivers on the installation disks.

The RNDIS protocol has a number of possibly desirable features not shared with other protocols:

- Uniform command and data encapsulation
- Designed to allow binding to other media than USB (e.g. IEEE 1394)
- Use with other networking protocols, such as ATM

The need to support (in principle) connection-oriented networking protocols (i.e. ATM) meant that MS couldn't use the much simpler Ethernet Networking Control Model defined as a part of the Communications Device Class by the USB Forum (see next section). However, the ATM part of the RNDIS specification has now been withdrawn. The other motivation, eventually supporting IEEE 1394 (using the same upper layer protocol with a different binding) has not borne fruit either. IEEE 1394 Ethernet adapters exist but use a different protocol. So from our present vantage point it would appear that RNDIS is a historical accident.

Nevertheless, RNDIS remains the only Ethernet-like networking protocol for USB supported by Microsoft.

Although RNDIS is a Microsoft product, all products implementing RNDIS must carry device identification information (vendor ID's and product ID's) appropriate to the product. Therefore, each deployed RNDIS device needs to supply a driver installation kit for the desktop OS. This kit installs and registers the latest RNDIS drivers to that device's identification information.

RNDIS is a closed protocol. It has various MS peculiarities, and therefore there is no Linux or Apple host driver for it, either open source or from other parties. If there is an identified need to support other communities via Ethernet over USB, then the device should support both RNDIS and the CDC Ethernet Network Control Model (see next section). On better UDC's, it is possible to present both RNDIS and CDC as selectable configurations ensuring that RNDIS is offered first because the Microsoft implementation of RNDIS will not consider alternate configurations.

A final peculiarity of the Microsoft implementation of RNDIS is that the presence of alternate non-RNDIS configurations may cause the MS implementation to reject the device as unknown. Belcarra RNDIS/CDC avoids this on all sufficiently capable UDCs, enabling a device to use RNDIS with MS platforms and CDC with other platforms.

In summary, for a UDC meeting certain requirements (see the UDC Requirements section below), the Microsoft-supported RNDIS protocol can be implemented. In that case, Microsoft supplies a host-side USB driver stack for all of its operating systems since Windows 98. Belcarra has developed an installer which pre-installs these drivers and binds them to a specific product and vendor ID pair. The Belcarra RNDIS function driver (for the device) can simultaneously support RNDIS and CDC (see next section) on

most UDC's meeting the requirements of both protocols (see below for details). The CDC Ethernet protocol is needed when either of two conditions is met:

- 1) there is a need to support non-Windows hosts; and
- 2) The UDC does not meet the technical requirements of RNDIS.

3 WHAT IS CDC Ethernet?

The USB Forum has defined a subclass of Communications Device called the Ethernet Networking Control Model. This was intended to become the standard way that USB networking adapters communicated with host PC's. This is the preferred protocol for non-Microsoft operating systems, such as Linux and Apple OS/X.

In the simplest and preferred case, RNDIS is presented as the primary configuration and CDC/Ethernet as the secondary configuration.

Host software on non-Microsoft operating systems⁵ will select the CDC⁶ configuration automatically. The Microsoft RNDIS driver will ignore the CDC Ethernet configuration as long as RNDIS is offered as the primary configuration.

3.1 CDC Ethernet, SAFE and Real-world UDCs

Certain UDCs, however, cannot implement standard CDC. This includes some UDCs which can implement RNDIS.

In the real world, USB is not the focus of today's SOC's⁷, but just one of many interfaces. In fact, it seems to have been given the least attention both in terms of resources and testing. The result is that many of the older, popular SOC's cannot implement either of the industry standard protocols (RNDIS or CDC Ethernet) properly.

The CDC model is simpler than RNDIS but requires a feature from the USB subsystem not present in certain popular embedded controllers. In other controllers, the UDC/CPU interface can lose data. This could be detected during frame reception at each end (by both host and peripheral drivers) using the Ethernet CRC, but the following language from the CDC standard indicates that this is not acceptable:

"The Ethernet Networking Control Model is used for exchanging Ethernet framed data between the device and host. A Communication Class interface is used to configure and manage various Ethernet functions, where an "Ethernet Networking Control Model" SubClass code is indicated in the descriptor definition of its Communication Class interface.

⁵ Including Belcarra's usbdnet for Linux and Belcarra OS/X CDC driver for Apple OS/X

⁶ CDC means *Communications Device Class*. CDC is used as a shorthand for *CDC Ethernet*, which is in turn a shorthand for *CDC Ethernet Control Model*. All three terms are used in contrast to RNDIS, which is also technically a CDC protocol, albeit a proprietary one.

⁷ System On Chip – all or a majority of interfaces and basic system hardware requirements are implemented on a single substrate along with the main microprocessor.

A Data Class interface is used to exchange Ethernet encapsulated frames sent over USB. These frames shall include everything from the Ethernet destination address (DA) up to the end of the data field. The CRC checksum must not be included for either send or receive data. It is the responsibility of the device hardware to generate and check CRC as required for the specific media. Receive frames that have a bad checksum must not be forwarded to the host. This implies that the device must be able to buffer at least one complete Ethernet frame”

To cope with these and other limitations of certain UDCs, the SAFE protocol has been defined. This protocol makes minimum demands on the enumeration process of UDCs, and can use a CRC32 encapsulation of data frames, thereby detecting UDC/CPU data corruption. Therefore, several variants of CDC are supported to conform to the peculiarities of assorted embedded environments. These are unified in a *de facto* private protocol known as SAFE.

Third party host-side drivers are available for CDC and SAFE for the Windows platform. For certain UDCs, namely those with limitations and other problems, this is the only viable alternative.

For modern UDCs, RNDIS is a simpler choice since Microsoft makes the host-side RNDIS drivers available without charge.

There is currently no driver of any kind, whether CDC or RNDIS, available for the Apple Macintosh without charge.

The Belcarra RNDIS/CDC Network driver can support either CDC Ethernet or SAFE as the primary or secondary configuration.

3.2 The Data over Cable (DOCSIS) 1.1 Standard

The DOCSIS standard encourages without quite mandating USB as a method of access to CPE⁸ (i.e. cable modems) from small sites, citing, among other things as reason to favour the USB access method (as opposed to Ethernet):

“Automatic device identification, configuration, and mapping of device function to its software, further simplifying the installation process (Plug and Play)”

Regarding the matter of how to attach USB modems to a host PC, the DOCSIS standard says the following:

“.. all USB attached DOCSIS cable modems MUST be compliant with either the Ethernet Networking Control Model or the Abstract Control Model as defined in the USB Communication Device Class. If the Abstract Control Model is used, then the CM MUST exchange Ethernet frames over the Data Class Interface and the CM MUST implement a Remote NDIS driver. “

This is DOCSIS-speak for saying that it must be CDC Ethernet or Remote NDIS – no other choices. The limited UDCs mentioned above cannot implement a DOCSIS compliant cable modem.

⁸ Customer Premises Equipment

3.3 Host Class drivers for CDC Ethernet and SAFE

3.3.1 Linux operating system

The Linux host drivers began as support for various dedicated USB Ethernet peripheral adapters, such as the Pegasus. These have been developed in a variety of directions by different parties for different purposes. Belcarra recommends the use of usbdnet (see next subsection) for use with smart peripherals.

3.3.1.1 usbdnet

For testing and validation purposes, Belcarra maintains a reference host class driver for desktop Linux. This driver is known as usbdnet and is distributed as part of Belcarra's RNDIS driver package. This driver transparently supports CDC Ethernet as well as SAFE and some other variants.

Since this driver is very important for USB development, and since it is not (yet) included in major Linux distributions, Belcarra provides this driver in Belcarra CD Linux, built otherwise from the standard Debian archives (www.cd-linux.com). This is a special edition of Linux which is booted from a CD, does not require a hard disk and does not need installation. CD Linux can be used immediately after the CD boots. This makes it an ideal test tool. A White Paper on CD Linux is in preparation.

3.3.1.2 CDCether

This is another GPL host driver supporting some variants of CDC Ethernet. This driver is not supported by Belcarra but can be found in certain recent versions of desktop Linux. It is capable of recognizing and using a device using the Belcarra RNDIS/CDC software if configured as a CDC (not SAFE) device. It does not however support the SAFE CRC encapsulation.

3.3.1.3 usbnet

This driver, found in all present major versions of Linux is the common ancestor of the two preceding drivers. Its purpose is to support certain USB Ethernet adapters. It is mentioned primarily because it has as similar name to usbdnet and is *not* supported by Belcarra. This driver can however be configured to support a device using the Belcarra RNDIS/CDC software if configured as a CDC or SAFE device. Belcarra's usbdnet supports a command to turn on CRC encapsulation in the device, rather than assuming that the device supports it. Other than that, the two drivers are quite similar.

The other major difference is that usbnet is designed to support a wide variety of peripherals.

3.3.2 MS Windows™ Operating Systems

Certain UDC's do not meet the RNDIS technical requirements (see above). For such cases there are proprietary drivers for Windows (available from others) which implement CDC Ethernet and SAFE.

3.3.3 Macintosh OS/X

A CDC Ethernet/SAFE combination class driver for OS/X is under development by Belcarra.

4 UDC Requirements

This section details the specific requirements for the various USB Ethernet flavours

4.1 RNDIS Requirements

Endpoints: CONTROL, BULK-IN, BULK-OUT, INTERRUPT
Other: CONTROL-WRITE

The RNDIS protocol requires 4 endpoints: default (control), BULK IN, BULK OUT, and notification (INTERRUPT IN) endpoint. It also requires support on the CONTROL endpoint for a CONTROL WRITE operation (setup packet followed by additional data.)

The RNDIS protocol does not normally rely on PNP⁹ to select the driver. After PNP has detected a supported Vendor ID/ Product ID pair, a specific driver is loaded and that driver communicates with the device to do an *extended enumeration* over the default pipe (CONTROL endpoint) to set up the device (establish frame sizes, etc). This is done using a USB Vendor-defined CONTROL WRITE packet. This in turn depends on the INTERRUPT IN feature (see above) so that the device can notify the host of the availability of a reply. Some UDCs, even though they have an INTERRUPT IN endpoint, still block direct software access to the enumeration process and thereby prevent extended enumeration.

4.2 CDC Requirements

Endpoints: CONTROL, BULK-IN, BULK-OUT, INTERRUPT
Other: SET INTERFACE with non-zero alternate setting

The CDC protocol requires 3 endpoints: default (control), BULK IN and BULK OUT. It may optionally use a fourth endpoint: INTERRUPT. It also requires that the UDC support a SET INTERFACE operation with a non-zero alternate setting.

The CDC protocol implements two USB interfaces:

Communications
DATA

The Communications interface is associated with the (optional) INTERRUPT endpoint.

⁹ Plug-N-Play

The DATA interface has two settings. The default (alternate setting zero) setting has no endpoints. The second setting (alternate setting non-zero) has the two BULK endpoints and is used to implement data flow.

Therefore the UDC must support multiple interfaces and an interface with multiple settings.

4.3 *SAFE Requirements*

Endpoints: CONTROL, BULK-IN, and BULK-OUT

The SAFE configuration is designed to support devices that do not implement one of:

INTERRUPT endpoint
CONTROL WRITE
SET INTERFACE

5 Extended Capabilities of Belcarra RNDIS/CDC

The Belcarra USB Network driver is designed to cope with special situations that occur in smart peripherals

5.1 *Soft CRC*

Despite USB having a hardware CRC check, certain UDCs can still lose data both inbound and outbound at the CPU/UDC boundary (due for instance to faulty DMA flow). Belcarra therefore implements the SAFE protocol, which includes software CRC32, with very acceptable impact on overall link speed, ensuring end-to-end data integrity despite faulty hardware.

5.2 *Built-in DHCP*

The Belcarra Network driver can be optionally configured to provide basic DHCP service to the host PC. This can be better than running a DHCP server at the application layer because the DHCP information is supplied at module load time. Ordinary DHCP servers are not used to dealing with network interfaces which appear and disappear. The Belcarra solution is simpler and smaller making it suitable for embedded devices.

6 Taxonomy

The original network driver developed for HP and Sharp, and released under the GPL license, is net_fd. This driver used Version 1 of the USB Device Framework.

Belcarra has independently implemented its own network_fd, with comparable but extended functionality to net_fd, implementing all protocols except RNDIS. To further improve efficiency and also reduce the memory footprint of the drivers, all Belcarra Drivers use the revised Version 2 of the USB Device Framework.

Belcarra has implemented `rndis_fd` as a proprietary extension of `network_fd`. The same module supports any combination of RNDIS, CDC Ethernet, and SAFE, underlying hardware permitting. The RNDIS choice is always presented first because the matching protocol used by the Microsoft RNDIS drivers requires this.

7 CONCLUSIONS

For the normal case of supporting Windows users on a modern UDC, RNDIS is the right choice. The host-side drivers are supplied and supported by Microsoft for all of their various USB-capable operating systems. All you need is Belcarra's RNDIS extension to the USB function module. Belcarra supplies a driver installation kit for end-users to use.

Similarly, to support Linux and Macintosh users, Belcarra's RNDIS can supply CDC Ethernet as an alternate configuration to RNDIS. Belcarra's own forthcoming OS/X driver will provide a solution for OS/X users. We know of no other offerings for Macintosh users.

Finally, if the UDC cannot support RNDIS, then CDC Ethernet or SAFE must be used, along with a third-party Windows client driver solution.

Since the Belcarra RNDIS/CDC equipped device is loaded with both configurations (RNDIS/CDC or RDNIS/SAFE) it can be used with a Windows, Linux or Macintosh without reconfiguration. This specifically makes such user paradigms as syncing a PDA to a Windows PC at work and a Mac OS/X at home practical and easy.