

Address Management and Device Discovery

Overview

This paper describes how to build and manage a TCP/IP network for USB devices.

Ethernet-style networking can very simply be extended to attached USB devices. Belcarra's *USBLAN for Window's* built-in DHCP server for address assignment, management and device discovery solutions has been implemented to fit within the network paradigm requirements. *USBLAN for Window's* implements USB-IF (www.usb.org) Communications Device Class protocols (CDC-NCM, EEM, ECM and others) and will create an Ethernet-style networking segment for each attached USB device.

Networking over USB protocols such as CDC-ECM, Microsoft RNDIS, CDC-NCM, were originally designed to provide USB-mediated access to an external network, and address management was seen as a matter for the external network segment. On such external segments, a DHCP server was usually available.

Now, however, the great majority of USB devices using these protocols are smart devices which use a network style command channel, but do not provide access to an external network medium. Therefore, the network segment terminates at the device, and there are only two nodes on the network: the USB host (Windows) and the device. The DHCP service within Belcarra *USBLAN for Window's* is a special service for such network segments.

The following describes how the network layer (IP) addresses can be seamlessly set up on top of the link layer and some pointers and strategies on OEM software running on Windows co-operating with the device to enable network-based device discovery.

Link Layer

At lowest level, a Windows network interface is an NDIS interface, using Ethernet-like (IEEE 802.3) framing. The associated network segment consists of the new interface and the device. The two devices (Windows and the attached USB device) can send network (Ethernet/802.3) frames between each other but that traffic will not be routed or bridged to anywhere else (unless the user takes action to bridge the new interface with others). IP address assignment and management

For the device to be accessible via TCP/IP protocols such as HTTP, the network interfaces of the devices must also have IP addresses assigned to them, and this needs to match the address the device uses (be on a common subnet). If these addresses are pre-arranged (simple but inflexible), then the IPv4 properties of the new interface (the IP address and netmask) can be set to the values by hand (manual setup). Another procedure is to have the device operate a simple DHCP server which sends the same information to the Windows side. This does not require user intervention on the Windows side, but will fail in the case of address conflict from another device, especially if a second device of the same type is concurrently active.

Belcarra USBLAN for Windows DHCP server

To gracefully and transparently assign IP addresses to both the Windows interface and to the device, Belcarra's *USBLAN for Windows* has an optional DHCP server, visible only to the ends of the new network segment. The DHCP server is within the USBLAN driver, but does not inter-operate with

network facilities for other non-USB network interfaces, in particular other DHCP servers or clients that might be operating.

Automatic address assignment for concurrently active devices.

This feature allows up to 32 devices to be concurrently connected. Each device is assigned a tiny subblock as a private network consisting of the Windows interface, the DHCP server's address within that block, and the device. The rules of sub-networking say that line address and the broadcast address are also reserved, so that a 5-address sub-network is needed. So Belcarra's internal DHCP service uses a block of 8 addresses for each concurrent device.

Features of Belcarra USBLAN for Windows internal DHCP service

- Neither the device nor the Windows end-user needs to participate in device setup
- DHCP lease times are configurable. This means that after a device receives a DHCP lease (and associated IP address), and is subsequently disconnected and reconnected within the lease time, it can request the same IP address (without first waiting for a new DHCP Offer). If the lease time (by default ten minutes), this request will fail and the fall-back is to request a new DHCP lease (including an IP address).
- Up to 32 devices can be connected concurrently
- The Windows host can readily discover which device(s) are active (and by implication which previously active devices are absent) purely by network operations.
- The server is optional. Customer kits enable or disable the kits according to customer request. At runtime the server is enabled if an associated registry key is set. This can be changed with regedit. This is mainly for evaluation. In normal end-user operation, this key would not be changed (and therefore admin access would not be required).

As such this is a plug-and play solution -- no action on the device or on the Windows side is needed to maintain coordinated address assignment and device discovery.

The addresses used are subdivided from a block of 256 addresses which is configurable from the registry.

Details of the DHCP subnet setup

Since each end of the network segment, plus the DHCP server needs its own IP address, the DHCP server sets up each block of 8 addresses within the master block of 256 addresses as shown in Table 1 following.

TABLE 1

Offset	Name	Default first value
--------	------	---------------------

0	Line Address	169.254.1.0
1	IP address of Windows NDIS interface as assigned by USBLAN DHCP server	169.254.1.1
2	IP address assigned to device by USBLAN DHCP server	169.254.1.2
3	DHCP Server address visible to both device and Windows NDIS interface	169.254.1.3
7	Broadcast address within the CIDR/29 subnet	169.254.1.7

The first few device addresses

Device discovery is described in some detail below, but for many purposes, it suffices to note the list of possible device addresses, and in some cases it may suffice to investigate the first few slots

Table 2 following shows the first 8 DHCP address assignments, using the default base of 169.254.1.0

TABLE 2

Device index	Windows NDIS IP address	Device address assignment
0	169.254.1.1	169.254.1.2
1	169.254.1.9	169.254.1.10
2	169.254.1.17	169.254.1.18
3	169.254.1.25	169.254.1.26
4	169.254.1.33	169.254.1.34
5	169.254.1.41	169.254.1.42
6	169.254.1.49	169.254.1.50
7	169.254.1.57	169.254.1.58

The DHCP server normally uses the network mask 255.255.255.248 (first 29 bits 1, remaining 3 bits zero), but can allow devices the wider value 255.255.255.0.

Network-based Device Discovery

Given that one or more USBLAN network devices are attached, Table 2 above provides a way to search for such devices in the form of a short list of addresses. The base of the list is dependent on the single registry NETWORKBASE which is the line address of Device 0 (see Table 1). At any given time, some subset of those addresses will be active. The key to working with all these devices, without knowing which are working, is to send UDP packets to the possible addresses.

The list of possible network addresses, as returned by Windows network calls, will include non-USB network interfaces, but the address blocks will be inconsistent.

With this in mind, we can imagine a simple discovery process, as follows. The addresses shown are merely samples.

Process one on Windows

1. Get a list of the active network interfaces
2. For each interface matching the network base (others may be non-USB interfaces), add the interface to the multicast group
3. Send a multicast UDP request to address 226.1.1.1:2000. This will be sent to all active devices. In another thread, the Windows machine listens for responses to its requests. Since UDP is not reliable, a link may not respond immediately. Therefore, the Windows side can conclude after 2 or 3 failures that the device is not responding properly. Similarly, when compared to a previous list, Windows may conclude that a no-longer-responding device has been disconnected.
4. Branch off process two (see below)
5. Repeat periodically

Thread/process two on Windows

1. Bind to UDP port 2001. Listen for packets

Thread/process on device

1. Add the USB interface to the multicast group
2. Bind to UDP port 2000.
3. Listen for requests
4. When a request comes in (see above), discover the regular unicast address of the request, and send a UDP reply to port 2001 at the source (see above)
5. Go back to item 3