WireShark Dissector for CobraNet Packets

Ver 1.0 28 November 2018

Installation

Like any dissector, this is "installed" by copying the Lua file (dissector_cobranet.lua) to your local user storage, i.e.,

C:\Users\<login name>\AppData\Roaming\Wireshark\Plugins

For example, if your login name is "jdoe", put it in the folder: C:\Users\jdoe\AppData…

If Wireshark is already running, such as with a file loaded, you can have it bring in the new plugin by going to the **Analyze** menu and selecting **Reload Lua Plugins**, or hitting the shortcut keys **Ctrl+Shift+L**.

Basic Dissector Information/Usage

Once loaded, display of CobraNet packets in the top window change so that in the protocol column, instead of showing a number (0x8819), it now says "CobraNet". And, in the info column, it shows what type of packet it is: Beat, Reservation or Audio.

No.	Time	Source	Destination	Protocol	Length	Info
38	0.000000	PeakAudi_05:8d:d3	PeakAudi_fe:58:23	CobraNet	160	Audio
39	0.000977	Audiosci_00:12:b3	PeakAudi_ff:ff:00	CobraNet	116	Beat
40	0.000976	PeakAudi_05:8d:d3	PeakAudi_fe:58:23	CobraNet	160	Audio
41	0.000000	Innovati_62:02:a5	PeakAudi_ff:ff:01	CobraNet	140	Reservation
42	0.000977	Audiosci_00:12:b3	PeakAudi_ff:ff:00	CobraNet	116	Beat
43	0.000000	PeakAudi_05:8d:d3	PeakAudi_fe:58:23	CobraNet	160	Audio
44	0.000976	Audiosci_00:12:b3	PeakAudi_ff:ff:00	CobraNet	116	Beat
45	0.000000	PeakAudi_05:8d:d3	PeakAudi_fe:58:23	CobraNet	160	Audio
46	0.001954	Audiosci_00:12:b3	PeakAudi_ff:ff:00	CobraNet	116	Beat

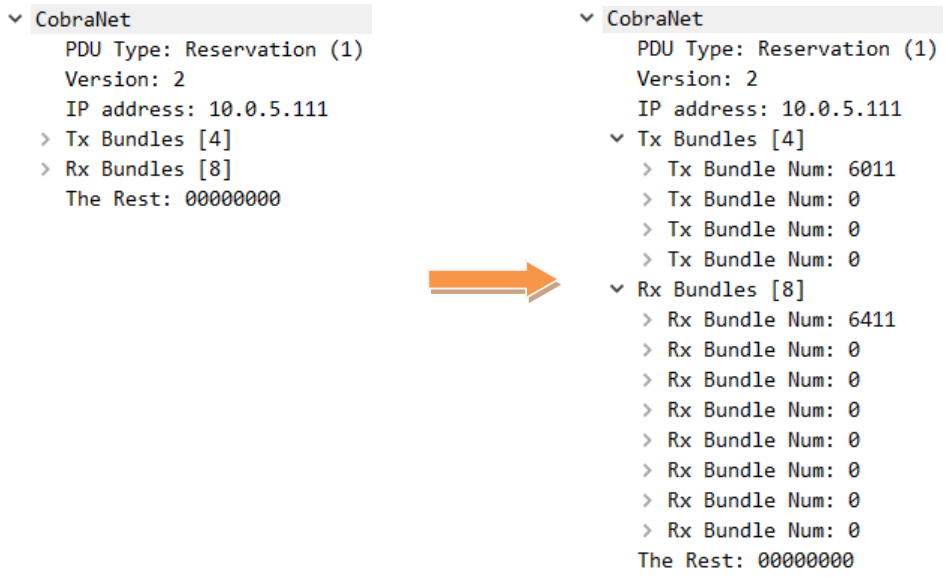
The CobraNet packets are further dissected and displayed in the **Packet Details** pane (be sure it is enabled in the **View** menu). At first one may see something such as below.

```
> Frame 1: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0
> Ethernet II, Src: Audiosci_00:12:b3 (00:1c:f7:00:12:b3), Dst: PeakAudi_ff:ff:00 (01:60:2b:ff:ff:00)
> CobraNet
```

The last line says "CobraNet" and can be further expanded by clicking on the greater-than symbol (>) beside it (circled in screenshot). This shows the dissection for the packet such as the example below.

```
> Frame 1: 116 bytes on wire (928 bits), 116 bytes captured (928 bits) on interface 0
> Ethernet II, Src: Audiosci_00:12:b3 (00:1c:f7:00:12:b3), Dst: PeakAudi_ff:ff:00 (01:60:2b:ff:ff:00)
v CobraNet
    PDU Type: Beat (0)
    Version: 2
    Cycle Number: 42436
    Cycle Rate (pkts/sec): 750
    Conductor Priority: 228
    Reservation Renewal Interval (cycles): 6300
    The Rest: 0000000000001c4000602b058dd300602b062c7f001cf700...
```

The Reservation dissection has some sub-trees in it due to the variable size of the transmitter and receiver lists. These may appear with the greater-than symbol, and can be opened up by clicking on the symbols.

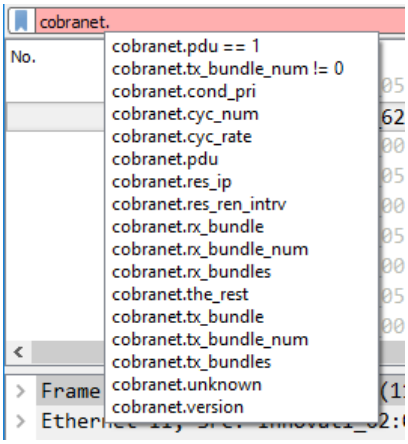


More Advanced Usage

With the CobraNet dissector installed, one can use this to do additional filtering and searching in Wireshark. For example, to find all assigned CobraNet Transmitters on the network, one would use the filter expression:

(cobranet.pdu == 1) && !(cobranet.tx_bundle_num == 0)

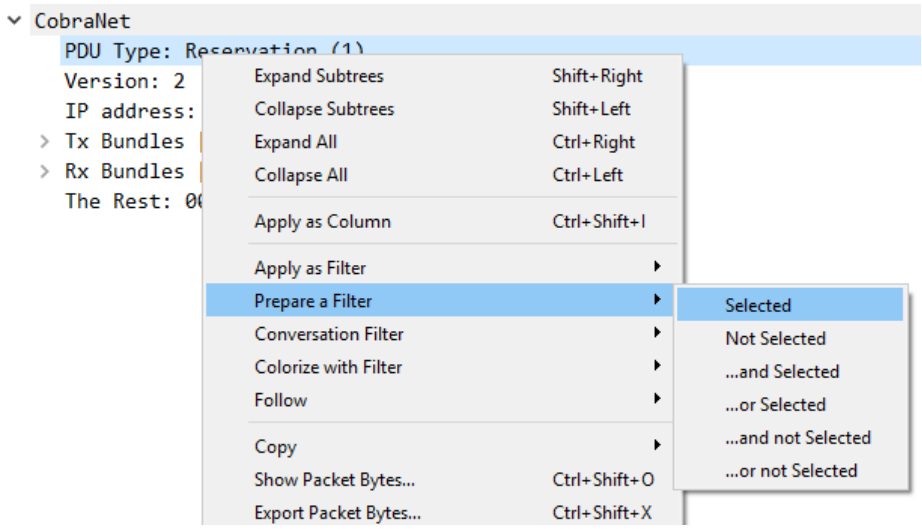
That is non-zero TX bundle number and packet type of reservation. Audio packets also have transmitter bundle numbers in them, hence the need for the second term in the expression in case there are also audio packets in the capture.



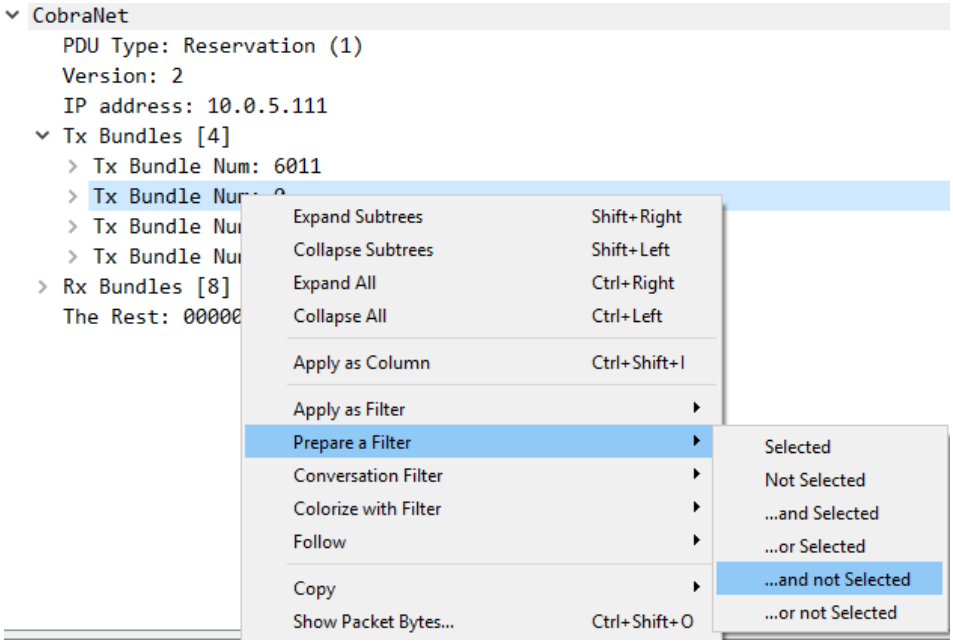
Okay, so how does somebody know how to build such filter strings? There are two ways, with Wireshark helping you along the way. Method 1 is to start typing in the filter box. Once you get "cobranet." (must include the period), Wireshark will display a prompt of available fields, such as shown at right. This prompt includes both recently used expressions (like the "cobranet.pdu == 1" shown), followed by all the available fields in the CobraNet dissector. One can either continue typing one of the expressions/fields or simply click on the item in the prompt. (If the prompt box goes away due to loss of window focus, you can bring it back by deleting and re-typing the period.)

To complete the filter, one may have to use parentheses around each expression and join them with logical operators like "and" and "or" or the programmer versions: && and || (Wireshark accepts either for of logical operators).

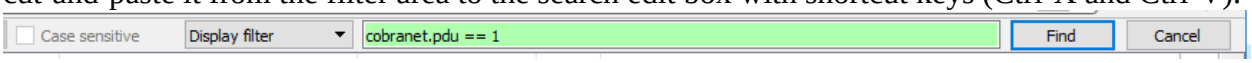
Method 2 is to right-click on some item in the CobraNet dissection that appears in the Packet Details window and select either **Apply as Filter** or **Prepare a Filter** from the pop-up menu, followed by ones choice in the sub-menu. For example, to create the filter shown previously, one would click on the PDU type field in the Packet Details, Prepare a Filter and Selected.



Then, expand the Tx Bundles sub-tree and do the same right-clicking on a zeroed out TX Bundle, such as below. One can manually edit/tweak the filter settings in the edit box now if desired, or hit the go button (right arrow) or [Enter] key to have Wireshark apply the filter.



The same expressions can be used in the search (find) function as well if one leaves it set to the default mode of “Display Filter”. Wireshark does not have the aids to build up the search expression that it provides for the filter, so a good strategy would be to prepare a filter and then cut-and-paste it from the filter area to the search edit box with shortcut keys (Ctrl-X and Ctrl-V).



Lua Dissector File

In the event these instructions get separated from the .lua file needed for Wireshark, the file can be reconstructed by copying the text below to a text editor like Notepad or Notepad++ and saving it as dissector_cobranet.lua (although the name is not critical, only the extension).

```
-- Wireshark dissector for CobraNet protocol (ethertype == 0x8819)
-- Decodes all of reservation packet and accounts for different lengths of
transmitter/receiver lists for different technology/chips
-- Also decodes fields in top of Beat packet and top of Audio packet
--    Ken Tench <ken.tench@atlasied.com>
-- This version is an expansion upon the incomplete version started by
--    Eliot Blennerhassett <eblennerhassett@audioscience.com>

do
   -- Create a new dissector
   COBRANET = Proto ("cobranet", "CobraNet")
   local cobranet_ethertype = 0x8819

   -- Create the protocol fields
   local pdus= {[0]="Beat", [1]="Reservation", [0x10]="Audio"}


   local f = COBRANET.fields
   f.pdu = ProtoField.uint8 ("cobranet.pdu",  "PDU Type", nil, pdus)
   f.version = ProtoField.uint8 ("cobranet.version",  "Version")

   f.res_ip = ProtoField.ipv4 ("cobranet.res_ip",  "IP address")
   f.cyc_num = ProtoField.uint16 ("cobranet.cyc_num", "Cycle Number")
   f.cyc_rate = ProtoField.uint16 ("cobranet.cyc_rate", "Cycle Rate
(pkts/sec)")
   f.cond_pri = ProtoField.uint8 ("cobranet.cond_pri", "Conductor Priority")
   f.res_ren_intrv = ProtoField.uint16 ("cobranet.res_ren_intrv", "Reservation
Renewal Interval (cycles)")
   f.tx_bundles = ProtoField.string("cobranet.tx_bundles")
   f.tx_bundle = ProtoField.bytes ("cobranet.tx_bundle",  "Tx Bundle")
   f.tx_bundle_num = ProtoField.uint16 ("cobranet.tx_bundle_num",  "Tx Bundle
Num")
   f.rx_bundles = ProtoField.string("cobranet.rx_bundles")
   f.rx_bundle = ProtoField.bytes ("cobranet.rx_bundle",  "Rx Bundle")
   f.rx_bundle_num = ProtoField.uint16 ("cobranet.rx_bundle_num",  "Rx Bundle
Num")
   f.unknown = ProtoField.bytes ("cobranet.unknown",  "Unknown")
   f.the_rest = ProtoField.bytes ("cobranet.the_rest", "The Rest")

   function tx_bundle(buffer, subtree, n)
          local tx = subtree:add_le(f.tx_bundle_num, buffer(2 + n * 6, 2))
          tx:add(f.tx_bundle, buffer(n * 6, 6))
   end

   function tx_bundles(buffer, subtree, num_tx)
          local n
          local tx_tree = subtree:add(f.tx_bundles)
          tx_tree:set_text("Tx Bundles ["..num_tx.."]")
          for n = 0, num_tx-1, 1 do
                tx_bundle(buffer, tx_tree, n)
          end
   end

   function rx_bundle(buffer, subtree, n)
          local rx = subtree:add_le(f.rx_bundle_num, buffer(2 + n * 10, 2))
```

```
        rx:add(f.rx_bundle, buffer(n * 10, 10))
    end

    function rx_bundles(buffer, subtree, num_rx)
        local n
        local rx_tree = subtree:add(f.rx_bundles)
        rx_tree:set_text("Rx Bundles ["..num_rx.."]")
        for n = 0, num_rx-1, 1 do
            rx_bundle(buffer, rx_tree, n)
        end
    end

    -- The dissector function
    function COBRANET.dissector (buffer, packet, tree)
        -- Adding fields to the tree
        local subtree = tree:add (COBRANET, buffer())
        local offset = 0
        local n
        local pdu_buf= buffer (0, 1)
        local pdu = pdu_buf:uint()
        local sect_len
        local num_txrx

        packet.cols.protocol:set("CobraNet")
        packet.cols.info:set(pdus[pdu])
        subtree:add (f.pdu, pdu_buf)
        subtree:add (f.version, buffer (1, 1))
        offset =  2
        if pdu == 0 then
            subtree:add_le(f.cyc_num, buffer(4,2))
            subtree:add_le(f.cyc_rate, buffer(8,2))
            subtree:add(f.cond_pri, buffer(10,1))
            subtree:add_le(f.res_ren_intrv, buffer(16,2))
            offset = 18
        end
        if pdu == 1 then
            subtree:add(f.res_ip, buffer(10, 4))
            -- Read length of section from next header
            offset = offset + 12
            sect_len = buffer(offset,1):uint()
            num_txrx = (sect_len -1 ) / 3
            tx_bundles(buffer(offset+2), subtree, num_txrx)
            -- Read length of section from next header
            offset = offset + (2*sect_len)
            sect_len = buffer(offset,1):uint()
            num_txrx = (sect_len -1 ) / 5
            rx_bundles(buffer(offset+2), subtree, num_txrx)
            offset = offset + (2*sect_len)
        end
        if pdu == 0x10 then
            subtree:add_le(f.cyc_num, buffer(4,2))
            subtree:add_le(f.tx_bundle_num, buffer(6,2))
            offset = offset + 6
        end
        subtree:add (f.the_rest, buffer(offset))
    end
    ether_table = DissectorTable.get ("ethertype")
    ether_table:add (cobranet_ethertype, COBRANET)
end
```