# UDP Broadcast on Machines with Multiple Network Interfaces

By Mircea Neacsu

A few days ago our tech support department ran into an interesting problem that I think is worth mentioning.

The customer (USCOE Galveston) has a new Win7 machine with two network interfaces. On one interface they have the Applanix POSM/V and on the other interface they have the ODOM CV-3 echosounder. As long as the POSM/V is disconnected everything works fine. As soon as the POSM/V is connected, the ODOM does not receive Start/Stop command and annotation strings from HYPACK.

The communication with the ODOM echosounder is done using broadcast UDP packets on port number 1600 for reading and 1601 for sending commands to the echosounder. The driver always uses the local broadcast address 255.255.255.255.

A bit of Googling let me discover a few strange facts about broadcasting on Windows® machines:

- **On XP machines, the broadcast packets are sent on all interfaces but with the IP address of the first interface.** For example, imagine this network configuration (order is important):
  - **Adapter 1:** IP address `192.168.0.1`
  - **Adapter 2:** IP address `10.0.0.1`
  - **Adapter 3:** IP address `172.17.0.1`

  When a broadcast is sent, the following packets will be sent (with source and destination IP addresses):
  - **On adapter 1:** `192.168.0.1` => `255.255.255.255`: OK
  - **On adapter 2:** `192.168.0.1` => `255.255.255.255`: bad source IP address
  - **On adapter 3:** `192.168.0.1` => `255.255.255.255`: bad source IP address

  The bad source IP address is important only if the other side must reply to the sender because the return packet might not be properly routed. The machine receiving the broadcast on network 10.x.x.x will probably have no idea how to send to 192.168.0.1.
- **On Windows® 7 machines, broadcast seems to be controlled by the IP route table**.

***TABLE 1.*** *IPv4 Route Table*

```
===========================================================================
    Active Routes:
    Network Destination        Netmask          Gateway       Interface  Metric
              0.0.0.0          0.0.0.0   10.202.254.254      10.202.1.2     286
              0.0.0.0          0.0.0.0      192.168.0.1    192.168.0.3      10
           10.202.0.0    255.255.0.0         On-link         10.202.1.2     286
           10.202.1.2  255.255.255.255         On-link         10.202.1.2     286
      10.202.255.255  255.255.255.255         On-link         10.202.1.2     286
            127.0.0.0        255.0.0.0         On-link          127.0.0.1     306
            127.0.0.1  255.255.255.255         On-link          127.0.0.1     306
      127.255.255.255  255.255.255.255         On-link          127.0.0.1     306
          192.168.0.0    255.255.255.0         On-link       192.168.0.3     266
          192.168.0.3  255.255.255.255         On-link       192.168.0.3     266
        192.168.0.255  255.255.255.255         On-link       192.168.0.3     266
            224.0.0.0        240.0.0.0         On-link          127.0.0.1     306
            224.0.0.0        240.0.0.0         On-link       192.168.0.3     266
            224.0.0.0        240.0.0.0         On-link         10.202.1.2     286
      255.255.255.255  255.255.255.255         On-link          127.0.0.1     306
      255.255.255.255  255.255.255.255         On-link       192.168.0.3     266
      255.255.255.255  255.255.255.255         On-link         10.202.1.2     286
===========================================================================
```

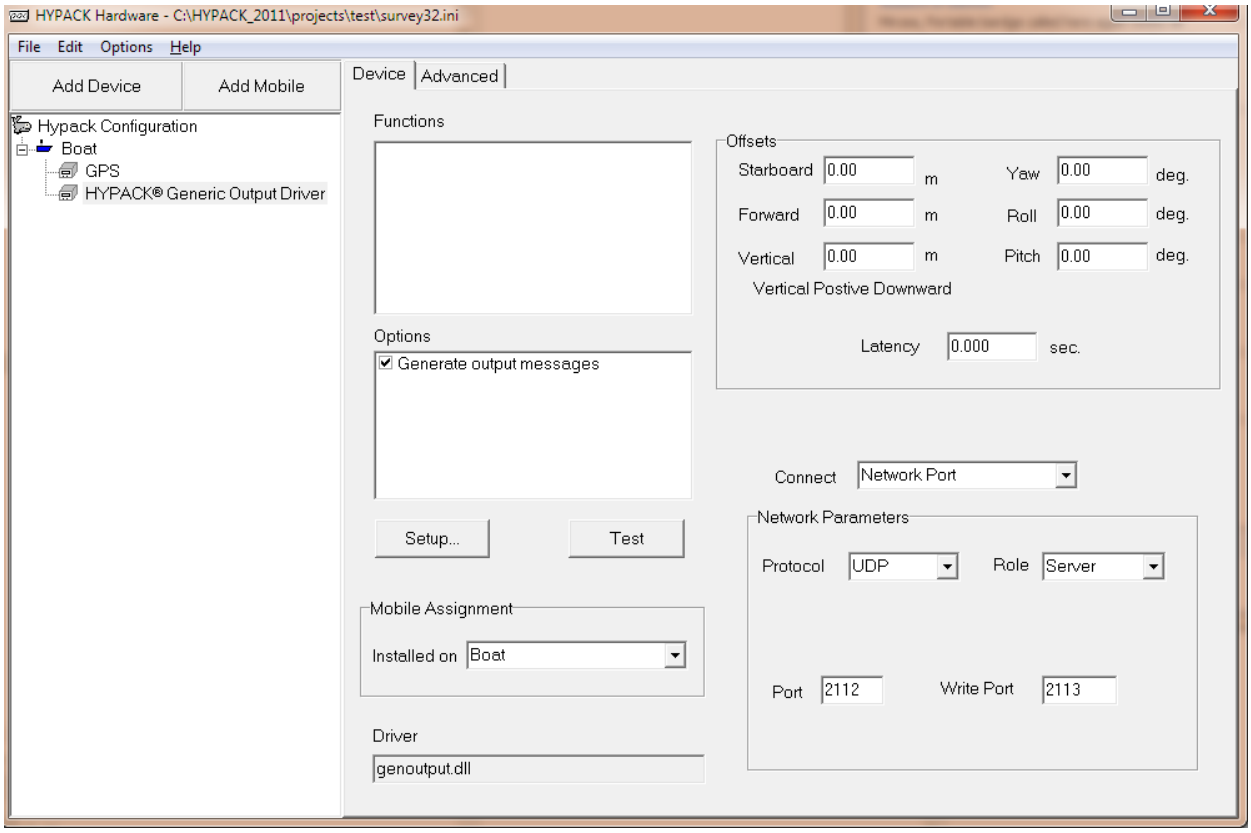
> The 255.255.255.255 routes control broadcast packets. In this situation, broadcast packets will be sent via the `192.168.0.3` because it has the lower metric... but not to the other interfaces.

(As I said before, I didn't discover this myself; it is based on the information from http://social.technet.microsoft.com/Forums/en-US/w7itpronetworking/thread/72e7387a-9f2c-4bf4-a004-c89ddde1c8aa/.)

The gentleman who researched this, went ahead and created a small program that listens on the local interface 127.0.0.1 and re-sends any broadcast message to all interfaces except the preferred one. The program can be downloaded from http://winipbroadcast.e-t172.net/releases/WinIPBroadcast-1.2.exe.

To test how this works, I setup a hardware configuration where one device is set to broadcast UDP messages:

***FIGURE 1.*** *Configuring UDP Output*




If I start Wireshark and capture the packets sent on the default interface, I see the expected output:

***FIGURE 2.***

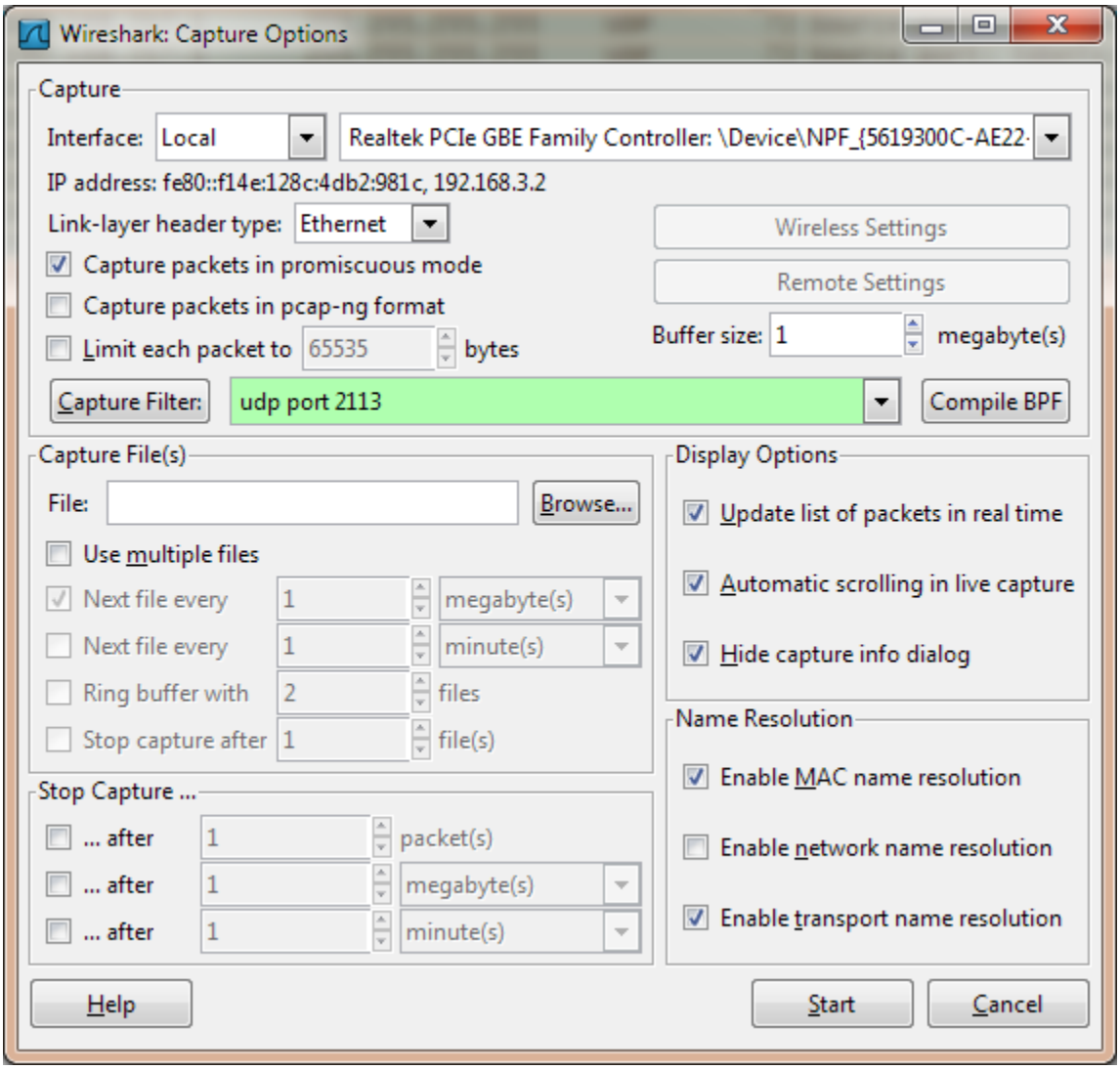
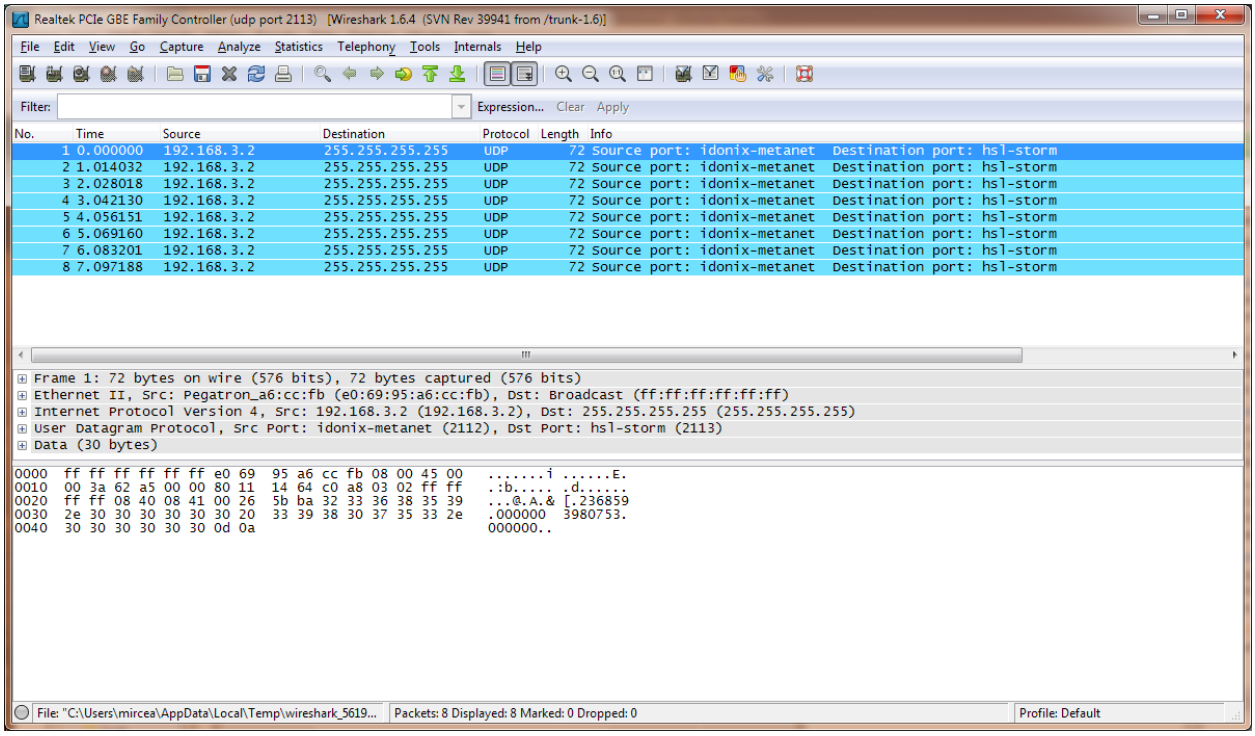

Wireshark: Capture Options
Capture
Interface: Local
Realtek PCIe GBE Family Controller: \Device\NPF_{5619300C-AE22-
IP address: fe80::f14e:128c:4db2:981c, 192.168.3.2
Link-layer header type: Ethernet
Wireless Settings
Capture packets in promiscuous mode
Remote Settings
Capture packets in pcap-ng format
Buffer size: 1 megabyte(s)
Limit each packet to 65535 bytes
Capture Filter: udp port 2113
Compile BPF
Capture File(s)
File:
Browse...
Use multiple files
Next file every 1 megabyte(s)
Next file every 1 minute(s)
Ring buffer with 2 files
Stop capture after 1 file(s)
Display Options
Update list of packets in real time
Automatic scrolling in live capture
Hide capture info dialog
Name Resolution
Enable MAC name resolution
Enable network name resolution
Enable transport name resolution
Stop Capture ...
... after 1 packet(s)
... after 1 megabyte(s)
... after 1 minute(s)
Help
Start
Cancel

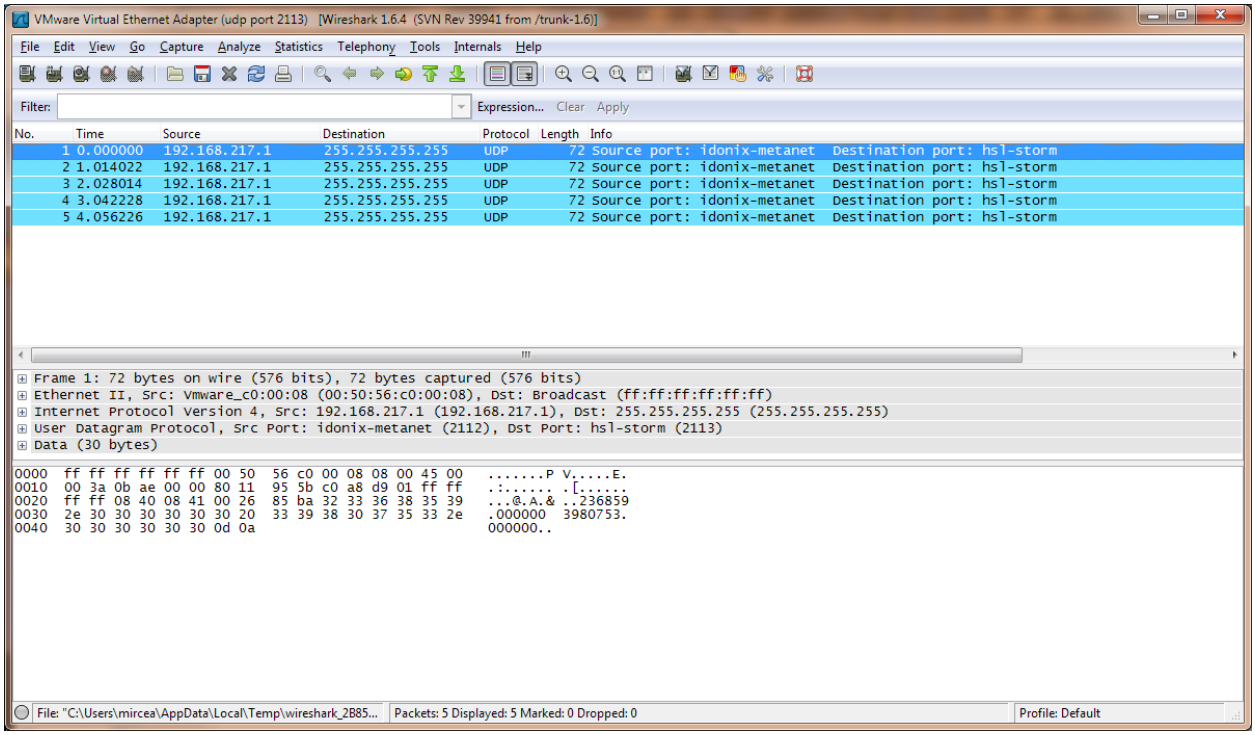
***FIGURE 3.*** *Expected Output*




If I start the same capture on another interface, I don't receive anything.

Now I start the WinIPBroadcast program and, lo and behold, the packets start should up:

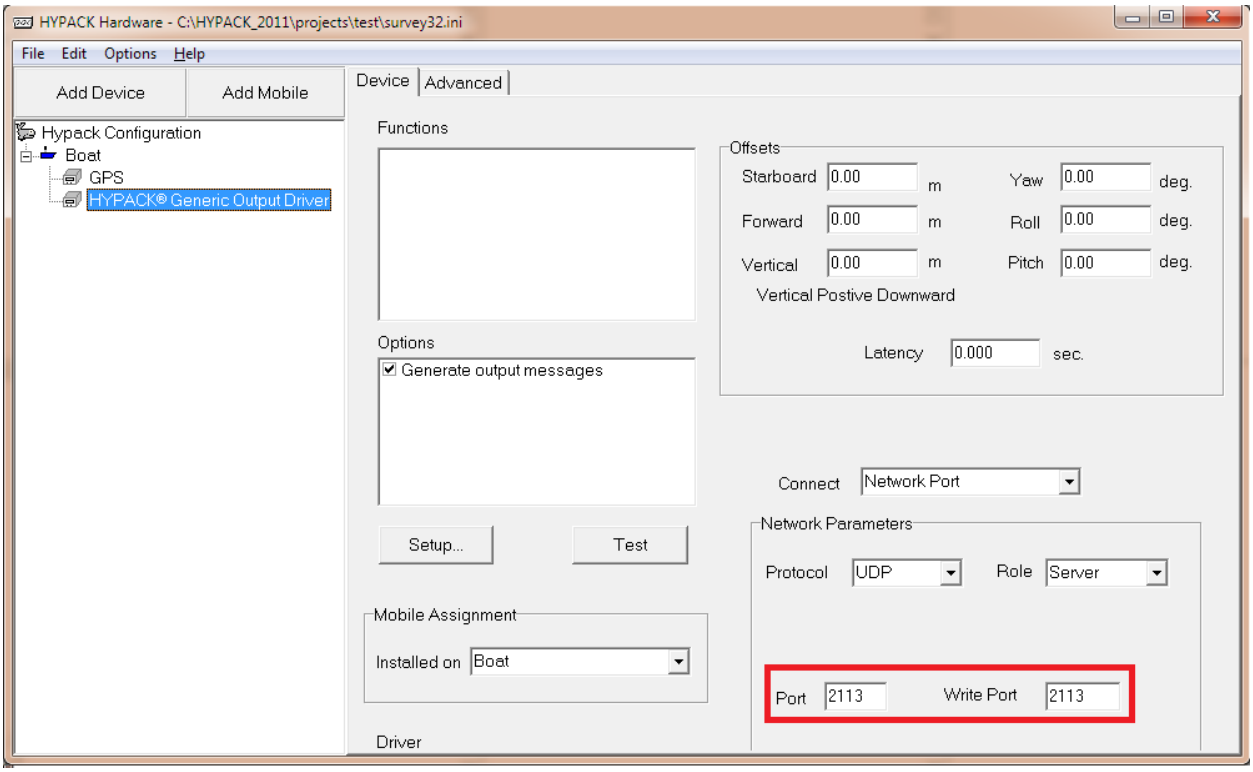
***FIGURE 4.*** *the source IP address has changed to reflect the IP address of the second NIC card*



Notice how the source IP address has changed to reflect the IP address of the second NIC card.

Is everything perfect? Unfortunately not! If the source port number is the same as the destination port number (notice the same port number for read and write) then WinIPBroadcast does not send the packets on the other interfaces; they get sent just on the primary NIC.

***FIGURE 5.*** *Read and Write Port Have the Same Address*




I still don't know why people would use multiple network interfaces on their data acquisitions machines. It makes for a more complicated setup with more wires and more chances of messing something up for the doubtful benefit of not using a $50 network switch. However, if this is your configuration, be forewarned that you might run into trouble and you might have to use the small program mentioned before to solve problems with broadcast datagrams.