OCKAM
INSTRUMENTS,INC.

The Ockam UDP Broadcast

The “UDP Broadcast” is the mechanism whereby the Ockam bus is connected to an Ethernet network. The Eye™ PDA has used this mechanism for many years to get Ockam data and control the instrument system. Recently, Expedition has adopted it as an optional data source in addition to the serial port. With third parties involved, it’s time to publish the specification.

What it Is

The Ockam UDP broadcast carries several channels of data and status information, mostly outbound from the Ockam system. These include the complete Ockam display channel (the data your displays listen to), the NMEA data stream and several other items.

Inbound data includes the keyboard channel (i.e. user commands, average setting commands etc.) and a client registration process.

Communication is carried out using the *User Datagram Protocol* (or **UDP**; see http://en.wikipedia.org/wiki/User_Datagram_Protocol), broadcast to all nodes on the local Ethernet – wired and wireless. The main advantage of a UDP broadcast is a one-to-many topology. Another advantage is that no handshaking is required. If somebody on the local network wants to receive the data, all he has to do is open the correct UDP port, and the data starts arriving.

How it Works

UDP ‘datagrams’ are indivisible globs of arbitrary data with a maximum length of about 1400 bytes. The sender puts the data together, specifies the length of the datagram, and addresses it to a particular recipient, or to all recipients (a ‘broadcast’). He must always define a port number that will receive the data – there is no broadcast to ‘all ports’. UDP broadcasts also do not pass thru gateways, and therefore do not go out onto the wider internet.

The magic of Ethernet is that one data stream involving port ‘a’ has no effect on other data streams using port ‘not a’. As far as the machines using port ‘a’ are concerned, they are alone on the Ethernet. To be sure, there can be so much traffic on the Ethernet that things get bogged down for everybody. However, on the average boat, this is unlikely to happen.

A disadvantage of UDP is the ‘unreliable’ nature of transmission. If a UDP datagram gets lost or mangled, neither the sender nor recipient gets notified. The

TCP protocol offers reliable communication (i.e. automatic retransmission of missing data) at the price of some considerable overhead. And classical TCP is point-to-point – no broadcast. Because missing the occasional Ockam bus packet is not serious (the next one will be along in ¼ second), UDP broadcast makes sense. And the UDP broadcast carries a sequence number, so communications reliability can be monitored.

The port number

As you can see from the above, the port number implicitly defines the stream. The default port number for the Ockam UDP broadcast is 5005. The server sends UDP packets to 'everybody listening on port 5005'. Each server and client must agree on this number for the broadcast to work. For each server and client, there is a setup item defining this port, so in case of conflict (or more interestingly, in case of more than 1 boat's data stream), the port number can be changed.

Traffic going the other way (from client to server) uses a port number one greater than the server port. In other words, the server listens for inbound UDP packets to port n+1 (i.e. 5006 or whatever the broadcast port is plus 1).

Ockam UDP broadcast packets

All Ockam UDP broadcast packets consist of a fixed-length header (16 bytes) plus a variable length payload (but not over 1384 bytes). The header contains the following:

- An identifier "OK1<0>" (4 bytes).
- A protocol revision number, currently 1 (2 bytes).
- A message type identifier. Currently there are 16 types (2 bytes).
- A sequence number, maintained by the server (4 bytes).
- 4 spare bytes.

Here is a summary of the currently defined message types. Unless specifically mentioned, the direction is server to client(s). All payloads are ASCII.

Message ID	Description
0 OKUDP_STAT	Ockam device registration. The data pairs an IP address and a name. Any OKUDP_REGCLI (type 6) packets are rebroadcast, and the server adds a separate packet with its own ID. These packets tend to occur about 1/second. Example: "192.168.1.108,LANbridge 0.0,index.html". If the device has a web page, it is included as the 3rd item.

1 OKUDP_BUS	Ockam data verbatim off the bus. This is the instrument display output. For details on the Ockam bus and what it contains, see http://www.ockam.com/docs/howbusworks.html. These packets occur at 4/second (or 8/second for the T1 in high rate mode). Example: ",<cr><lf>6<0>T15:11:37<0>B6.73<0>…"
2 OKUDP_NMEA	Complete NMEA sentences, sent as soon as received. The <cr><lf> is replaced by <null>. These packets come at the same rate as the GPS sends them. Example: "$GPGGA,123456,4114.125,N,07300.123,W…"
3 (client to server) OKUDP_CMD	Keyboard commands to the server. Example: "K1=1.06"
4 OKUDP_VARDAT[1]	Current definitions for each Ockam bus tag, e.g. defines tag 'B' as boatspeed. The payload is tag, long name, short name, type, granularity, maximum length, # decimal digits. Items are comma separated. Example: "+,Wndspd App Axial,Vax,1,1.0,5,1,0".
5 OKUDP_AVGS[1]	The current value for all instrument averages. Example: "16.0,6.0,4.0,10.0,8.0,8.0,10.0,10.0,…"..
6 (client to server) OKUDP_REGCLI	Clients of the UDP broadcast identify themselves to the server by occasionally sending this message. The server retransmits this message on UDP_STAT when received. Example: "192.168.1.108,LANbridge 0.0,index.html". LANbridge repeats these frames on type 0.
7 OKUDP_ENUMCLI[1]	Enumerated client list. The server maintains a list of clients and enumerates them with this message. Example: "0,192.168.1.108, LANbridge 0.0,index.html"
8 OKUDP_ALERT	Alert string. Never implemented.
9 OKUDP_BIFINFO[1]	List of all BIFs. Example: "n,e,Name1,..,NameN<0>" e:0=off; 1:on.
10 (client to server) OKUDP_BIFCTRL)	Manipulates the BIF functions. List with BIF#, tagDecimal, CurAlt, Enabled. Example: "3,51,0,1" turns on BIF 3 on tag '3', option 0.
11 (client to server) OKUDP_LOGSTRING	Inserts the string with a time stamp into the current log file.

12 OKUDP_RCIMAGE[1]	Defines the parameters of the current race course. There are 5 subheadings: Header 0,Rev,Flags,LastWp,LastRt,CrsName WpEnum 1,n,Rev,Icon,CircType,WpName, Lat, Lon, Radius, StartAngle, EndAngle RtEnum 2,Rt[0],...,Rt[LastRt-1] CurLeg 3,n<0> (Wpt[Rt[n]] is next mark) Laylines 4,0<0> (None) 4,1,Lat1,Lon1,ET1 4,2,Lat1,Lon1,ET1,Lat2,Lon2,ET2
13 OKUDP_CMDACK	Acknowledges OKUDP_CMD. Sequence number =sequence number of CMD.
14 OKUDP_JGRPINFO[1]	Enumeration of the jumbo group info. Sent round-robin 1/second or when changed. Example: "<N>,<Name1>,...,<NameN>,<nCurGrp>"
15 (client to server) OKUDP_JGRPSEND	Causes the server to send the jumbo settings defined in the specified group. Example: "1"

[1] Status messages (VARDAT, AVGS, ENUMCLI, BIFINFO, JGRPINFO and RCIMAGE) are sent round-robin by group at 4/second rate. In other words, all VarDats are sent, then the avgs, then all the clients, etc. If any item changes, it is broadcast immediately without disturbing the status enumeration. LANbridge does not provide these frames.

How to play with the Ockam UDP broadcast

The Ockam UDP broadcast is sourced either by the OckamSoft 4 driver (i.e. from a serial port input to your PC), or the 051L LANbridge. The LANbridge provides Ockam data in lieu of serial ports, so no onboard PC is required (see http://www.ockam.com/LANbridge/index.html for details).

You can play with the UDP broadcast for free.Install OS4 (http://www.ockam.com/os4/index.htm) and start the driver.

Open the driver (icon in the 'tray'), and enable the simulator to provide data. Download (http://www.ockam.com/docs/UDPmonInstall.exe) and run UDPmon which displays the contents of the UDP broadcast.