ETHERNET/IP PROTOCOL NETWORK MESSAGE HOUSKEEPING

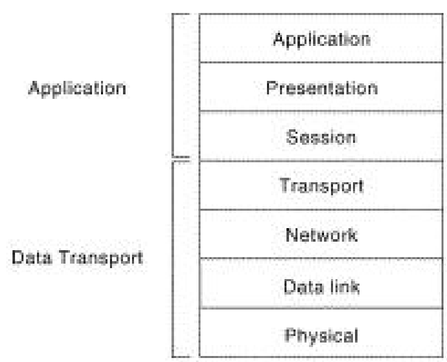
By Erik Syme, ProSoft Technology

The following information is designed to assist in the configuration of an EtherNet/IP network over a radio network.

EtherNet/IP has two standard types of messages. The first type of messaging, PCCC (PC cubed), is a DF1 message that has been adapted to the EtherNet/IP specifications. These messages are unconnected, and because they do not have to go through the connection manager this type of messaging can reduce the traffic on an EtherNet/IP network.

CIP is another type of messaging reference below. This type of messaging is connected messaging, and must go through the connection manager. Because of this connection manager, each time a MSG instruction is enabled in a processor, a FWD Open and FWD Close request must be sent on the EtherNet/IP line.

Remember that since EtherNet/IP uses TCP/IP for it's transport layer, all EtherNet/IP messages are connected messages. Each type of messaging, PCCC and CIP, must be connected on the TCP/IP layer. CIP messages go through an additional connection at the Application Layer, after they have already established the connection at the Session Layer. The following shows the 7 layers of TCP/IP communications:



All of the messaging that we are talking about is on the Application layer. We will briefly discuss the connection of EtherNet/IP, however this is not the area of focus here. We are only talking in the reduction of the traffic at the Application layer (shown above).

The following will illustrate from the opening TCP/IP sequence on through the completion of one MSG instruction from a Control Logix processor with a 1756-ENBT to a Compact Logic L35E processor. The below MSG example is a PCCC style message.

167	6.248608	192.168.0.191	192.168.0.118	TCP	1026 > 44818 [SYN] Seq=3107975613 Ack=0 Win=4096 Len=
168	6.249725	192.168.0.118	192.168.0.191	TCP	44818 > 1026 [SYN, ACK] Seq=2379589053 Ack=3107975614
169	6.250334	192.168.0.191	192.168.0.118	TCP	1026 > 44818 [ACK] Seq=3107975614 Ack=2379589054 Win=
170	6.251658	192.168.0.191	192.168.0.118	ENIP	List Services (Req)
172	6.255892	192.168.0.118	192.168.0.191	TCP	44818 > 1026 [ACK] Seq=2379589054 Ack=3107975638 Win=
173	6.255897	192.168.0.118	192.168.0.191	ENIP	List Services (Rsp)
174	6.256587	192.168.0.191	192.168.0.118	TCP	1026 > 44818 [ACK] Seq=3107975638 Ack=2379589104 Win=
175	6.257426	192.168.0.191	192.168.0.118	ENIP	Register Session (Req), Session: 0x00000000
176	6.259013	192.168.0.118	192.168.0.191	ENIP	Register Session (Rsp), Session: 0x0A020300
177	6.260405	192.168.0.191	192.168.0.118	ENIP	Send RR Data (Req), Unconnected Send, , Unkno
178	6.268234	192.168.0.118	192.168.0.191	ENIP	Send RR Data (Rsp), Unknown Service (4b)

By looking in the far right column you can see that for the completion of one MSG instruction and how it appears on the Ethernet Line or Radio Network.
On PCCC the sequence for the messaging is as follows:

SYN - client to server
SYN ACK - server to client
ACK - client to server

At this point the TCP/IP socket is open.

List Services - client to server
List Services Response - server to client

Register Session - client to server
Register Session Response - server to client

At this point the session has been registered and data can be exchanged between the two devices.

Send RR Data - client to server
Send RR Data Response - server to client

Data has now been exchanged between the two devices.

Now we will look at a sample of a CIP message. Below is the traffic on the wire for a CIP style EtherNet/IP message:

409	28.359827	192.168.0.191	192.168.0.118	TCP	1030 > 44818 [SYN] Seq=4183431613 Ack=0 Win=4096 Len=0 MSS=1460 WS=0 TSV=519750 TSE
410	28.360826	192.168.0.118	192.168.0.191	TCP	44818 > 1030 [SYN, ACK] Seq=3454917053 Ack=4183431614 Win=4096 Len=0 MSS=1460 WS=0
411	28.361598	192.168.0.191	192.168.0.118	TCP	1030 > 44818 [ACK] Seq=4183431614 Ack=3454917054 Win=4096 Len=0
412	28.362833	192.168.0.191	192.168.0.118	ENIP	List Services (Req)
413	28.364470	192.168.0.118	192.168.0.191	TCP	44818 > 1030 [ACK] Seq=3454917054 Ack=4183431638 Win=4096 Len=0
414	28.365532	192.168.0.118	192.168.0.191	ENIP	List Services (Rsp)
415	28.366181	192.168.0.191	192.168.0.118	TCP	1030 > 44818 [ACK] Seq=4183431638 Ack=3454917104 Win=4070 Len=0
416	28.367083	192.168.0.191	192.168.0.118	ENIP	Register Session (Req), Session: 0x00000000
417	28.369678	192.168.0.118	192.168.0.191	ENIP	Register Session (Rsp), Session: 0x0A020700
418	28.369816	192.168.0.191	192.168.0.118	ENIP	Send RR Data (Req), Forward Open
419	28.377610	192.168.0.118	192.168.0.191	ENIP	Send RR Data (Rsp), Forward Open
420	28.385491	192.168.0.191	192.168.0.118	ENIP	Send Unit Data (Req), CONID: 0x00150341, Unknown Service (4c)
421	28.388322	192.168.0.118	192.168.0.191	ENIP	Send Unit Data (Rsp), CONID: 0x00254102, Unknown Service (4c)
422	28.397955	192.168.0.191	192.168.0.118	ENIP	Send RR Data (Req), Forward Close
423	28.406438	192.168.0.118	192.168.0.191	ENIP	Send RR Data (Rsp), Forward Close[Unreassembled Packet]

Once again, you will see the familiar opening sequence, this time using CIP messaging. By looking at the far right hand column you can see the following:

SYN - client to server
SYN ACK - server to client
ACK - client to server

At this point the TCP/IP socket is open.

List Services - client to server
List Services Response - server to client

Register Session - client to server
Register Session Response - server to client

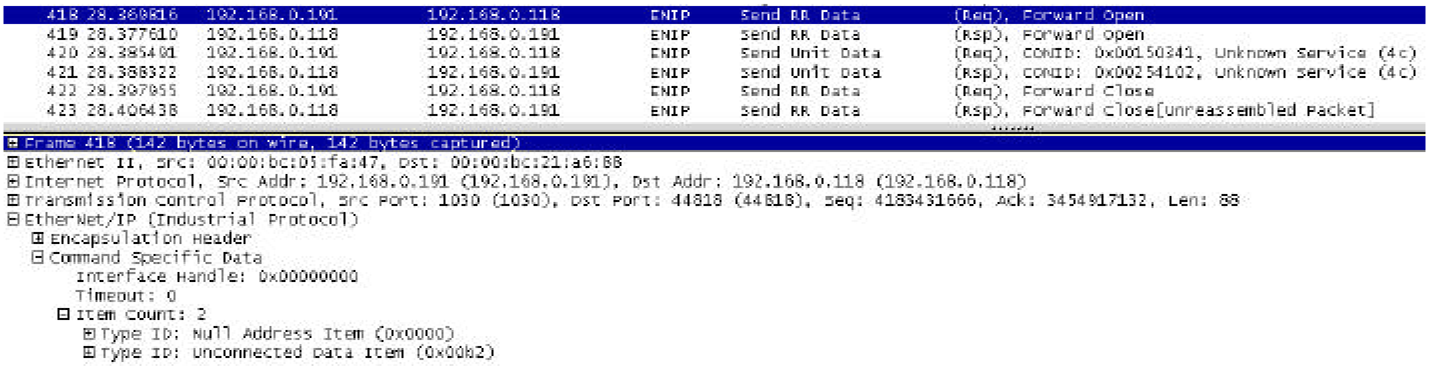
At this point the session has been registered and data can be exchanged between the two devices.

Every data request with CIP style messaging will now require 6 messages instead of the 2 messages (Send RR Data) as was seen with PCCC style messages.

Now instead of just 2 Send RR messages to exchange data via PCCC messages, a Send RR Data is used for a FWD Open (Forward Open), then a Send Unit Data is used to exchange data. After this a Send RR Data is necessary for a FWD Close (Forward Close) message.
Below is the sequence:

Send RR Data (FWD Open Request) - Client to Server
Send RR Data (FWD Open Response) - Server to Client
Send Unit Data (Request) - Client to Server
Send Unit Data (Response) - Server to Client
Send RR Data (FWD Close Request) - Client to Server
Send RR Data (FWD Close Response) - Server to Client
On CIP messaging, data has now been exchanged.

If we look at the FWD Open Request and Response, FWD Close Request and Response we can see the amount of data that is added on the EtherNet/IP network using the CIP messaging. Below is the FWD Open message:



```
418 28.368816   192.168.0.191   192.168.0.118   ENIP   Send RR Data     (Req), Forward Open
419 28.377610   192.168.0.118   192.168.0.191   ENIP   Send RR Data     (Rsp), Forward open
420 28.385401   192.168.0.191   192.168.0.118   ENIP   Send Unit Data   (Req), CONID: 0x00150341, Unknown Service (4c)
421 28.388322   192.168.0.118   192.168.0.191   ENIP   Send Unit Data   (Rsp), CONID: 0x00254102, Unknown Service (4c)
422 28.397055   192.168.0.191   192.168.0.118   ENIP   Send RR Data     (Req), Forward Close
423 28.406438   192.168.0.118   192.168.0.191   ENIP   Send RR Data     (Rsp), Forward Close[Unreassembled Packet]

⊞ Frame 418 (142 bytes on wire, 142 bytes captured)
⊞ Ethernet II, Src: 00:00:bc:05:fa:47, Dst: 00:00:bc:21:a6:88
⊞ Internet Protocol, Src Addr: 192.168.0.191 (192.168.0.191), Dst Addr: 192.168.0.118 (192.168.0.118)
⊞ Transmission Control Protocol, Src Port: 1030 (1030), Dst Port: 44818 (44818), Seq: 4183431666, Ack: 3454917132, Len: 88
⊟ EtherNet/IP (Industrial Protocol)
   ⊞ Encapsulation Header
   ⊟ Command Specific Data
       Interface Handle: 0x00000000
       Timeout: 0
     ⊟ Item Count: 2
        ⊞ Type ID: Null Address Item (0x0000)
        ⊞ Type ID: Unconnected Data Item (0x00b2)
```

In looking at the size of Frame 418, you will see that it is “142 bytes on wire”.

Below is the sizes of each of the Frames shown above:

Frame 418 (FWD Open Request)	- 142 Bytes
Frame 419 (FWD Open Response)	- 124 Bytes
Frame 422 (FWD Close Request)	- 118 Bytes
Frame 423 (FWD Close Response)	- 108 Bytes
	492 Bytes

By using PCCC style messaging instead of CIP style messaging, there can be a reduction of 492 bytes per MSG instruction in a Control Logix or Compact Logix processor.

By reducing the amount of Bytes needed for data exchange you can improve overall throughput on your network, and a reduction of total traffic.