

Native EMS to Field Device Integration Using DNP Over Ethernet

Presented by Joe Orth, Tacoma Power

**By Dave Eastcott, Bow Networks & Joe Orth, Tacoma Power
DistribuTech, 2002**

Abstract

DNP has been used for many years for serial communication between different manufacturers Energy Management Systems (EMSs) and Remote Terminal Units (RTUs) and is recommended by the IEEE for communication between substation devices. DNP 3.0 (Distributed Network Protocol version 3.0) documents related to serial communication are well thought out and the protocol is widely used throughout the world. In 1998 a paper was developed by the DNP Technical Committee that described DNP use over TCP (Transport Control Protocol) and UDP (Universal Datagram Protocol). There are several methods by which this integration can be achieved – one is by using third party terminal servers that wrap and unwrap DNP making the LAN (Local Area Network) / WAN (Wide Area Network) implementation superficial to the EMS and end device equipment. Another method (that is described in this document) is to have DNP over Ethernet implemented directly on the EMS front end processor and field devices. There are also variants of these approaches which will be touched on briefly. In this innovative project, Tacoma Power (TPWR) worked with three suppliers to integrate Tacoma Power's EMS and RTU data acquisition system utilizing Tacoma Power's wide area network (WAN). What originally appeared, on the surface, to be a simple task was actual quite complex and time consuming. The joy of being one of the first organizations to implement this type of native integration was also special. Due to issues not being clearly defined, various vendor specific issues, the utility was forced to act as the integrator and had to fill gaps and navigate a minefield of contractual and technical issues. The EMS is currently in Factory Acceptance Testing and is expected to be implemented at the utility site and integrated with both serial and Ethernet RTUs in the 4th quarter of 2001. At the time of writing this abstract, the system is functional on the test floor and the test includes 20 Ethernet RTUs and 52 serial RTUs. This paper discusses the steps that Tacoma Power went through in developing and managing the integration of this equipment to yield a successful implementation, the benefits gained by the use of DNP over Ethernet, and the shift to the Ethernet or "IT-Centric" environment. It also provides some dialog on the use of UDP and TCP in the data acquisition environment and includes future direction for substation equipment vendors that could provide additional benefits. This information should benefit utilities facing similar integration issues by raising thought provoking questions and sharing practical experience.

Either author welcomes comments and questions, and can be contacted at:

Dave Eastcott
Bow Networks
200, 550 71 Avenue SE
Calgary, Alberta T2H 0S6
Phone: 403/253-8433
FAX: 403/253-8979
Email: david@bowssoft.com

Joe Orth
Tacoma Power
3628 S. 35th Street
Tacoma, WA 98409
Phone: 253/502-8785
FAX: 253/502-8446
Email: jorth@ci.tacoma.wa.us

Background

Bow Networks is an Automation Communications company, specializing in the development of real time communication solutions for SCADA, EMS and Substation Automation products and systems. Working on a variety of hardware platforms, Bow provides the software solutions that connect legacy devices to today's network centric operations and decision support systems.

Bow Networks was founded in 1986, and over the past 15 years has become a recognized leader in providing mission critical software product development to OEM's (Original Equipment Manufacturers) and utility companies in the Electric Power industry. Many of the industry's most successful systems employ Bow technology for communications.

In September 2000, Tacoma Power (TPWR) and ABB Network Management entered into a contract for an EMS. Since 1994, Bow Networks has worked under contract with ABB Network Management and since that time Bow has developed and maintained the front end processor data acquisition software called TIE (Telemetry Integration Environment).

Tacoma Power is a municipal utility serving multiple jurisdictions in Pierce County, in Washington State, serving about 200 square miles and approximately 200,000 customers. Tacoma is about 30 miles south of Seattle.

In Tacoma Power's control area there are 4 major substations, 2 of which have 230 kV inter-ties with the BPA. Major substations are interconnected by a combination of 110 kV and 230 kV transmission lines. A 110 kV sub-transmission system distributes power from the major substations to Tacoma Power's unit substations. The typical unit substation has a 110 kV – 12 kV transformer with four, 12 kV feeders. In addition, there are also several double-ended bus substations. Tacoma Power has 60+ unit substations, 7 remote hydro plants and switchyards. Tacoma Power generates approximately half of the power consumed by our customers in these hydro facilities that are owned and operated by Tacoma Power.

It should be noted that almost all of the information contained herein has been presented to the DNP Technical Committee and summarily dismissed as not significant enough to require any modification or clarification to existing documents. A proposal to modify the port 20000 rule discussed herein was also dismissed. It is just a matter of time before it becomes readily apparent that these and some related issues need to be addressed. It is the intention of the authors to bring these issues into the open for discussion and to the attention of the end users, particularly as utilities move towards IT-centric environments and for end users to inform their vendors of what they want in the future.

Tacoma Power Wide Area Network (WAN)

In 1998, Tacoma Power launched Click! Network, a telecommunications venture to provide cable TV, high speed Internet access and other communication services for Tacoma Power customers. The Click! Network system uses fiber loops with coax cable to reach customers from the fiber loops. In parallel with the fiber optic portion of Click! Network is a separate system that is used by Tacoma Power, called the PASS (Power Administration SONET System). The

PASS uses Nortel's J-Mux (Jungle-Mux) equipment, which is installed in each substation and Tacoma Power's corporate office. Nortel equipment is also used for Click! Network but has separate termination equipment. The PASS does not include any coaxial cable. PASS is currently built out to about 70% of locations in the service territory.

The PASS implementation essentially divides Tacoma Power's service territory into 4 hubs which are in or near Tacoma Power's 4 transmission (major) substations. The PASS links these locations with the corporate complex using OC-3 circuits. From each of these, OC-1 fiber optic loops extend into the service territory. Not all substations are "on the loop", most are taps that are several blocks long (see Fig.1 on the last page – Tacoma Power PASS Architecture). Within the PASS System, 10 MB (Megabyte) is allocated for SCADA (Supervisory Control and Data Acquisition) Communications use. There are some high speed data links for SCADA use but most data acquisition is done over the 10 MB WAN. The term WAN is used to describe the geography. The reality is that this is essentially a 10 MB flat (single collision domain) LAN.

Several options were identified while designing the SCADA equipment architecture, choosing the desired SCADA functions and integrating the communication architecture for use by SCADA over the PASS. Point-to-point (dedicated links) versus Ethernet were the top options for SCADA use of the PASS. At the time, dedicated links were much more difficult to operate and maintain than they currently are, due to modifications by Nortel (now owned by GE), and as a result TPWR did not pursue that option any further. The J-Mux system includes an option for Ethernet cards. The system provides inherent routing functions (for Ethernet packets) using MAC addresses of equipment connected to the WAN, thus eliminating the need to install and manage routing equipment and sub-LANs. Basically, the J-Mux system is the proverbial communication cloud. That is, what goes in one location (say, as Ethernet) is bundled, delivered to the other side of the cloud, unbundled and delivered to the desired equipment (as Ethernet). What goes on in the cloud is not Ethernet. The J-Mux equipment is also suitable for use for relaying and other more demanding applications.

Based on some theoretical work and testing, an initial decision was made to have 4 data concentrators (one at each of the 4 major substations) that would report data over 56 kbit high speed serial links with the EMS. Those 4 major substation RTUs would gather data from neighboring/downstream RTUs using Ethernet. However, as TPWR worked through its EMS procurement, it became apparent that it would be possible to eliminate the intermediate "sub-masters" (over the long haul saving a lot of work) and integrate directly with the EMS using DNP over Ethernet. So we headed off in this direction and included DNP over Ethernet (both TCP and UDP) in our contract. We later realized that we were blazing a trail, that TPWR staff would have to gain much more expertise in this area to sort things out. Essentially, TPWR would have to act as the prime contractor between vendors to realize this vision and to ensure that any unexpected gaps would be filled.

Native vs. Non-Native

One question that had to be answered was how to accomplish this. There are two high level options: one to procure the EMS, have DNP over Ethernet implemented on the EMS front end processor (we'll call this the native solution) or two, TPWR could use terminal servers to

encapsulate the DNP in Ethernet and essentially steer clear of the integration issues (non-native solution). Both are viable options. Non-native is particularly appealing if a company has a legacy EMS for which adding DNP over Ethernet would be difficult, undesirable or costly. Several utilities have successfully implemented this type of system. Depending on the communication system available this option may also be a superior choice. However, in TPWR's case, the terminal servers would be an unnecessary piece of equipment in the data acquisition system deemed undesirable. Since TPWR already had a PC based system running DNP over UDP, and TPWR also had very favorable experience with this protocol and its robustness, it became TPWR's protocol of choice. TPWR had also reaped the benefits of working in an Ethernet environment, such as being able to trouble shoot equipment remotely, remote testing by forcing points and other items that eliminated substantial "wind-shield" time for TPWR's engineers and technicians. This was accomplished by being able to establish telnet sessions between PCs in our corporate complex and RTUs on the PASS LAN. Labor savings have typically been 2-8 hours per week. Other potential benefits of going to the Ethernet environment are still unrealized, as TPWR has to wait for RTU vendors to change their platforms to take advantage of well established technology (such as SNMP). An excellent paper on this subject was presented by Rick Murphy of First Energy at the 2001 DistribuTECH Conference titled "A Utility Perspective on Substation Wide Area Network Access."

UDP vs TCP

The next question was DNP over UDP versus DNP over TCP. Previous experience with UDP lead TPWR in that direction, but there was some thought put into the decision.

Of course, as you know, the DNP Data Link was designed for point to point communication links and wasn't designed to be a WAN protocol. Basically a DNP message consists of fragments which can generally range from 4 to 2048 (or higher) octets of user data. Each fragment is broken into frames which are generally 249 octets of user data. Each fragment of the message is given a sequence number (0 to 63), and each frame within a fragment is given an independent sequence number (0 to 15 for solicited or 16 to 31 for unsolicited messages). Having only a range of 16 for a frame sequence number is a little limiting for a WAN which in theory, can make it hard to tell if a response is a retry/duplicate or a leftover from a previous dialog. The chance of this happening is fairly remote, and to our knowledge, TPWR has never experienced this in several years of operation using UDP. As long as application layer confirms are used, acknowledgements or "acks" allow the sender to know which fragment was correctly received and thus act appropriately. This would resolve most packet issues in the WAN environment when using UDP.

The authors are not trying to argue for UDP vs TCP, but just relate some of the experiences that both have encountered. Anyway, it would be a schizophrenic argument anyway as one of the authors prefers UDP and the other TCP. The bottom line is that with appropriate safeguards in place, the reasons for using TCP versus UDP become less of an issue. For TPWR, both TCP and UDP were tested in the lab and found that to perform equally well due to the specific vendors' implementations (good work ABB Network Management, GE-Harris and Bow Networks). However, this will be discussed later. A note in favor of the work of TPWR's vendors is that they have implemented DNP over TCP and UDP intelligently.

A few words about TCP. There are three stages for a TCP connection:

- a) Establishing a Connection
- b) Exchanging Data
- c) Disconnecting.

a) The establish connection stage involves a few steps:

1. Resolve the target computer name to an IP (Internet Protocol) address value. This step is quick if the local computer contains in its host table the name and IP of the target computer, or it is in its ARP cache. If not, and the host computer has to contact a DNS (Domain Name Server) to obtain the IP for a given name, then there are a whole other series of transactions which need to occur (probably two or three).
2. Next, the IP layer in the computer attempts to resolve the IP address to an Ethernet card address (if the IP/Ethernet pair is not in its local cache). This requires a request ("who is IP...") and either a response or timeout to occur. If a timeout occurs, then the host assumes that the device is not on the local LAN segment, and tries to use the 'gateway' computer to forward its connection request.
3. A properly formatted IP packet is then sent to the target Ethernet address to request a connection. Three small messages are used to establish this connection. Basically, the first says "I want to Connect", the second says "OK, I accept your Request", the third says "I ACK your acceptance", and then data exchanges can occur. Note that there is other useful information in each exchange, specifically which socket each end wants the other to send subsequent packets to.

Excluding DNS lookups and assuming everyone is on the same physical segment, there probably would be 5 (2 + 3) messages exchanged before any application data began moving. If you add DNS into the equation, it becomes upwards of 5 plus 4 more messages, for approximately 9 messages.

b) The data exchange phase is not necessarily as bad as some would assume. Eventually, each transmitted message must have a corresponding ACK response, otherwise it will be transmitted again. However, TCP is a bit smarter than serial lines in that it uses a concept of a Sliding Window Technique. This means that the sender can transmit up to 'n' data packets before it receives an ACK to the first, but it must receive ACKS to all transmitted packages otherwise it will re-transmit. Also, if the network latency is long enough, when the Window limit is reached, the sender will stop transmitting until ACKS are received (or a timeout occurs) and then packets are re-transmitted.

c) Disconnecting a connection properly, involves 4 messages (2 for each direction):

- (1) the first computer says "I want to Disconnect"
- (2) the target computer responds "OK",
- (3) when the target computer is ready it will send to the first computer "I want to disconnect"
- (4) the first computer responds with "OK".

A few words about UDP. UDP is “connectionless.” That is, it doesn’t establish and manage connections like TCP (as described above).

With UDP, packets are ‘blindly’ sent from one port and IP combination (source) to a destination port and IP address. This is very similar to the operation of DNP on serial lines. Advantages include:

- less overhead than TCP (managing connections, etc.)
- protocol requires fewer resources
- probably better for large number of devices on LAN/WAN
- TCP does not support broadcast messages while UDP does

Downsides with the use of UDP include:

- messages may arrive out of order
- messages may take different paths
- higher likelihood that data may get lost
- problems with complex LANs/WANs with routers

When using DNP over UDP the DNP data link does provide additional reliability over UDP with the use of application layer confirms.

Use of the DNP broadcast address is applicable only to UDP and that capability is usually restricted to the sub-net within which the 'broadcaster' exists. As a rule, UDP broadcasts do not traverse segments (sub-nets), since intervening network equipment will block datagram transmission. Hence, the use of UDP within a larger network is very limited.

That being said, from TPWR’s perspective/business case, the single biggest problem with TCP occurs when an established connection is abnormally broken: one end or the other is turned off or resets. If the device failing was a single RTU/IED (Independent Electronic Device), then the host would probably not detect this for some time 16×4 seconds (16 is the usual number of retry attempts and 4 the usual number of retry timeouts used in the TCP layer), or about 64 seconds. Most systems allow you to adjust these values to optimize performance. However, the downside is that these values affect all application traffic, which is very important when the LAN/WAN is used by other unrelated applications. Parameter adjustment is not something which is normally done, since the existing applications may begin to operate abnormally, or, more likely, act abnormally at some future time when no one recalls that these parameters were changed. Conversely, if the host or LAN went away, the IED probably would not know about it for a minute and, if it is the type that will only accept a single session from any given host, it would probably refuse the host's connection attempts until it (the IED) timed out its previous connection with the host. UDP does not suffer from this. In addition to the wait for the RTU/IEDs to timeout, you would have to wait for the connections to be re-established by the EMS, which, for a 40-50 device LAN could easily add another minute (depending on several factors). The end result being at least 2 minutes without data acquisition being restored from all

devices. This doesn't work well for a data acquisition system – where even 30 seconds would be highly undesirable.

TPWR has experienced problems in the recent past with the SONET system going in and out of service repeatedly or losing synchronization. This mode of failure would be exacerbated if implemented with TCP. For example, the TPWR system employs a dual (hot standby) Ethernet Master design. Only one of the primary or secondary LANs poll at a time. When running TCP and a primary LAN failure occurs, the system fails over to the back up based on configured parameters, not socket timeouts and the failover time is the same as when using UDP. However, if the hot stand by master goes away (that is both primary and backup masters fail), then the primary TPWR system (assume it is functional again quickly) could experience IED busy problems mentioned earlier.

One aspect of the TCP protocol that gets a lot of attention is that it supposedly guarantees delivery of messages. In fact it does not. TCP is more robust and it transparently (to an application) retries transmissions, but it too has a limit that, when exceeded, will cause the loss of a data packet. This transparent retry mechanism of TCP is in fact a potential weakness when used with a data acquisition type system. The general TCP retry mechanism has two components: the time between retries and the number of retries before failing (reporting a transmit error) a transaction. For instance, suppose that the network becomes moderately to heavily loaded with traffic, and this triggers the TCP layer in some or all connected devices to begin going into their retry modes; you will see the nominal 150 ms exchange time (RTU response latency observed in TPWR's testing) will increase to possibly 64 seconds (4 seconds between retries, and with upward of 16 attempts before the application even knows there is a problem - $64 \text{ seconds} = 16 \times 4$). How many application programs set their Receive Timeouts to 70 seconds? So now the situation exists where the application (DNP) is retrying a command transmission long before the original message has even been dropped by the TCP layer. Be aware, the foregoing events occur on a moderate to heavily loaded system and may not be representative of what actually occur in other systems. There are a number of factors which need to be considered before drawing any absolute conclusions. One thing is certain though, with DNP and UDP, the application will know much sooner when problems occur (application level timeouts for responses or lack of application confirms) and begin to either report the problem or dynamically adjust its operation to compensate for these conditions.

Now assuming the following:

- the host computer just powered up (all ARP caches are empty),
- no DNS,
- using 150 msec message times (RTU response latencies were found to range between 150 and 400 msec depending on how busy the RTU was at the time it was polled – this time is the delay between when the RTU receives a poll from the Master to the time the RTU replies. We have assumed 150 msec for subsequent calculations),
- 50 devices other than the host,
- no other traffic on the LAN segment, and
- we are only establishing connections with no actual application data exchanges;

Using the above conditions we would approximate a $5 \times 0.15 \times 50 = 37.5$ seconds for basic connection time + 10% (computer internal overhead, another 3.75 seconds) for an estimated total of about 40 seconds to re-establish all 50 connections. If applications are allowed to exchange data as soon as their connection is established, and concurrently with the remaining connection establishments, the 40 seconds could easily expand to 60 plus seconds. If a DNS server is used, then the basic 40 seconds would increase to about 75 plus (9 x ...) seconds. For UDP these numbers would be 24 seconds without DNS and 40 seconds with DNS. Please note that if concurrent application data exchanges were allowed, then the numbers would increase.

Now add to the mix that once connections with more than one device have occurred, the likelihood that the network card's anti-collision mechanism will operate is fairly high, the more you have, the more likely re-transmissions and transmission back off's will occur. Consequently, as the number of devices increase, the effective utilization of the network segment reaches a certain threshold and then starts to decrease because of collisions and re-transmissions.

A final note: if the connection between the host and one or more of its "end devices" is outside the local segment, the RTU response latency time of 150 msec is going to likely be on the low side. A simple test is to use the 'ping' utility to ping various IP address and divide the reported time by two to get an approximate one way travel time. Also, ping a local segment device so that you can observe the relative differences.

During normal data exchanges, the TCP vs UDP issues appear to be usually inconsequential. It is during times of abnormal conditions, however, where one or the other protocol shines and the other can be a detriment. That choice is up to the end user.

There is one aspect of UDP/TCP over IP that is common: each protocol's packet header contains fields for both the Source IP and Source Port (the one sending the message) and the Destination IP and Destination Port number (who the message is for). This information is available to both participants in the conversation, TCP simply formalizes its use more than UDP does. Since this information is available it could be used. In the case of serial communications, the media source and destination address are the same, are the physical wires on which the message travels on. With serial communications, if I send a request to a device, I get the response back on the same port (physical wires). If I send a message via TCP, I get a response back which is addressed to my IP and Port number. However with UDP, the current DNP standard creates an exception. If I send a request to a device, the device always sends the response to a specific port address (port 20000) rather than the one I am using (unless I am using 20000), which can and has led to some interesting discussions and implementation issues that complicate things.

So, in summary, whether you use TCP or UDP, you probably always want to use DNP application layer confirmation turned on at the RTU/IED (that is the master is required to application layer confirm data it receives as a response from the RTU/IED).

Where Things Really Got Interesting

Where things really got interesting was when we wanted to take one vendor's RTU (under one contract with TPWR) and another vendor's EMS (under a different contract with TPWR) and

had them integrate using DNP over UDP (we also thought this could eventually be a good way to integrate other field devices with the EMS, but that thought got placed on the back burner). TPWR had thought this undertaking would be fairly simple and some DNP documents could be referenced and that would be it. This was not the case. In December of 2000, TPWR met with ABB Network Management, Bow Networks and GE-Harris to discuss how this was going to work and how we (the entire group) were to deal with the gaps in the DNP document.

The following is a summary of the issues we identified and felt needed to be dealt with. It is not our intent to find fault with anyone or the DNP Technical Committee about these issues but to see them resolved for the good of all. We know the DNP Technical Committee has had much discussion and anguish over these issues.

The First Problem Was with UDP Port Numbering Required by the DNP LAN/WAN Paper...

The DNP LAN/WAN paper ("Transporting DNP V3.00 over Local and Wide Area Networks," Version 1.0, dated December 15, 1998) requires that:

"All devices shall support TCP and UDP communications on port number 20000. This number has been registered with the IANA (Internet Assigned Numbers Authority) for use with DNP. All connection requests and all UDP data are sent to this common port number."

It is clear that the initial purpose of this statement does not allow for any other port numbers to be used when transmitting UDP data. It appears from discussion with the DNP Technical Committee that the purpose of this requirement is to ensure interoperability. However when this is applied to a large system, having all the data sent back to an EMS on a single port is not a good thing and not always desirable. This requirement in essence takes away the advantage of UDP being desirable for large LANs because the single port can only buffer so much data before data loss occurs. On another note, I have also come to understand that the TPWR system can still be considered compliant as long as it can be set up in the manner required by the above quoted statement and it can operate in any manner we see fit (which is non-compliant with the aforementioned statement). Either way, this statement can be misleading to other vendors and utilities. A possible solution that was proposed to modify the above statement to include either "by default" or "to demonstrate basic functionality devices must be initially configured to" was not given consideration.. As supporting information for the need to adopt this modification, please consider the following:

1. Limits on Security Solutions Part I - Security is repeatedly stated in writing as a primary concern for utilities and in regular letters from the National Security Council, EPRI is investing millions on this issue, etc. More and more IEDs are becoming accessible via 'public' networks: that is, either the public network acts as communications bridge between two 'private' networks (secure or insecure) or the IED(s) is on the 'public' network. Consequently security is a very big issue and may be addressed using a variety of approaches including hardware, software and a hardware-software hybrid solutions. The requirement to only use port 20000 for all TCP connections doesn't allow the simple approach of utilizing different ports for

establishing secure connections (such as SSL). Obviously, use of this and similar technologies for security imposes additional burdens on the communications requirements. While this concept is also applicable to other items, this paper just touches on SSL for connection authentication and data en/decryption as an example. Security is not a place to limit options.

Adding secure communications on top of TCP and below DNP is not really a problem, and the fact that you may have to have separate security contexts for each direction of the connection is not really a problem either (except for the person trying to maintain them). Communication can flow either way via either connection. Where the problem exists is on a mixed environment, when some devices need to be accessed using secure methods by other devices which exist on a secure network.

If all devices are required to initiate a connection via port 20000, how does the host easily differentiate which connections require security authentication and which ones do not? There are ways to accomplish this and still use only the 20000 port address, but this becomes messy and very dependent upon assumptions. Why the complexity? Why the limitation? Why can't the end user decide? It would be cleaner and safer if non-secure connections were initiated with (if desired) 20000 and secure connections with a port chosen by the user. There are other arguments which make sense from a host perspective which enables a degree of flexibility that the port 20000 only approach would prohibit.

2. Limits on Security Solutions Part II - There are other security issues, such as firewall rules and other items, which may mandate predefined port usage or the use of NAT (Network Address Translation) or IP Masquerading (a common technique for sharing a single world accessible IP address with a number of local computers) which make it impossible to comply with the port 20000 rule. NAT/IP Masquerading rely exclusively on the principle that a destination device will send its response to the IP and port number from which the request was sent. Consequently, this port 20000 requirement cannot be supported on all systems on which DNP over Ethernet may be used.
3. Problems with large LANs - The issue is really not the 'large' number of devices, in as much as it is the ability of the receiving device to service the incoming packets. Devices typically have a limited buffer space to deal with un-serviced incoming packets. Please note that most host stacks provide this buffer on a per socket (port) basis. If we use a single port at the host end, then there is nearly a 100% chance that packets will be discarded, when there is more un-serviced data than the buffer can hold. The host can issue a series of concurrent requests for data to different field devices, each of which could respond with a UDP packet of 1500 bytes, ($9k / 1.5k = 6$). With a buffer size of 9000 bytes, one can see that even with a small number of IEDs it would not take long before the host fell behind and began losing data. But using multiple ports and hence multiple buffers the likelihood of this occurring will drop dramatically.

TPWR recently ran a performance test with a substantial number of devices communicating with a single Master under a significant amount of activity. Each device had 400 analog points, or 25% which changed every two seconds and the devices were polled every 2 seconds). In this instance the LAN was flat and solely used by SCADA devices. These Ethernet RTUs were polled by one process using the 20000 Port address rule (UDP/IP). This process had to not only receive all the data but interpret the data. We found surprisingly long turn around delays in the remote devices (as the devices were busy) and we could only achieve our desired performance with 7 devices on one UDP port (100 to 450 msec delays for RTUs to respond). However, when we used UDP on two ports, we were able to have 14 devices on the LAN and still meet our performance requirements.

An increasing number of IEDs are being attached to LAN/WANs, both for inter-IED communications and access from at least one or more hosts. When communicating to a small numbers of connected devices, it appears the communication approach or methodology is not usually significant. However, communication performance with medium to large numbers of IEDs can be substantially affected, depending on the approach used. In this context, based on our experience, you can define a medium to large number of connected devices as 7 or more devices reporting 100 points every 2 seconds. Sequentially communicating with each device results in large loop times to get to all devices on the LAN. Unfortunately we found the time to complete a transaction for one “unloaded” device to be from 150 to 250 milliseconds (both in testing at the EMS vendor and in TPWR’s lab). Establishing parallel communications to all devices or groups of devices appears to consumes substantial resources at the host. However, parallel communications does have the benefit of using the bandwidth of the network more effectively, which yields the opportunity to have more devices on the network and better performance. However, because of the port requirement, a poll can be sent from any port on the FEP (Front End Processor) to any device, but the response from the device is required to go to port 20,000. Having one application at this port number that receives and processes data especially in an environment that has any kind of strict data update requirements is really only viable for low device count LANs. The number of devices on a LAN could be increased by vendors moving to improve response times, but that issue is for another forum. TPWR has clearly demonstrated the communication constraints in our lab and with our vendors. However, care must be taken with this process; it must have adequately large buffers, manage the buffers, not be CPU process starved, etc. Based on TPWR tests (stated in the previous paragraph) which utilized a fairly powerful UNIX workstation as the FEP, the single task methodology/approach doesn't work. Note that this approach becomes even less effective as the device latency issue is dealt with and more data arrives in a shorter time. Ultimately, this environment becomes risky - data is lost, remote device response timeouts occur because the data cannot be processed fast enough, FIFO buffer overflows, etc.

Moving to parallel communication increases the data acquisition performance of the network, which helps to compensate for device latency issues on a medium to large LANs. (obviously, it would be preferable for improvements in latency times/issues,

but, for the moment this is where things are at until field device vendors improve the power of their devices). There will always be large networks and it should be noted that the incremental cost of adding a device to an existing network is very small - adding another network, routers, etc. is higher cost. As customers, we have a set of parameters within which we must function (regulatory, corporate policy, vendor limitations, etc.) and need flexibility dealing with problems to meet our needs. Moving data in a timely manner is vital to a business in the utility environment. Specifically, this data can constitute a legal record of events that may be required to be provided in court or provided to regulatory agencies. Hence, utilities must take care to minimize the loss or potential loss of critical data. Explaining to lawyers, courts or regulators why data was not a pleasant prospect. Also, the prospect of facing an expert witness' testimony that responsible staff members did not take reasonable precautions to ensure data was not lost is equally unpleasant.

In large systems, particularly front end processors having only one task poll, parse and interpret data is not feasible. With serial communications, if a request is sent to a device, the response is back on the same port (physical wires). If a message is sent via TCP, a response is sent back which is addressed to the sender's IP and Port number. With UDP, the standard creates an exception. If a request is sent to a device, the device always sends the response to a different location (port 20000) rather than the one initiated the request (unless it was 20000). Why? Doesn't the LAN/WAN paper state a desire for similarity between serial and network DNP implementations? Having to have totally different front end code for UDP and TCP implementations not only doesn't make sense, it is inefficient and costs the end user.

As mentioned before, there is one aspect of UDP/TCP over IP that is common. Each protocol's packet header contains fields for both the Source IP and Source Port (the one sending the message) and the Destination IP and Destination Port number (who the message is for). This information is available to both participants in the conversation, TCP simply formalizes its use more than UDP does. Since this information is available it could be used. In the case of serial communications, the media source and destination address are the same, so are the physical wires that the message travels on. With serial communications, if a request is sent to a device, the response is returned on the same port (physical wires). If a message is sent via TCP, a response is sent back addressed to my IP and Port number. With UDP, the standard creates an exception. Hmmm...

The Next Problem Was How do we Manage UDP Port Numbering... Wild Sockets!

Not that we are lazy but TPWR didn't want to have to get into the business of managing port numbers. TPWR had a system wide plan for IP and DNP numbers already in place. If everyone agreed to operate contrarily to the UDP port number rule, port numbers could be managed by our EMS FEP by using wild sockets. In wild sockets, the EMS selects port numbers for the RTU to respond to. In a poll, the EMS sends its wild socket and IP address to the RTU at its IP and socket 20000 (all RTUs are configured to receive on port 20000). The RTU then sends its reply

to the senders (wild) socket and IP from the RTUs socket (20000) and IP. Further, we also agreed that the primary and backups Ethernet Masters would use the same socket numbers. The end result here was that everyone had to agree that we would operate outside the DNP LAN/WAN paper and the vendors wanted to also build in the flexibility to operate per the paper.

The Next Problem Was How Do We Handle Unsolicited Messages With UDP and TCP...

This issue was not addressed in the LAN/WAN paper. In a meeting of all the stakeholders it was easy to decide that it was logical to send unsolicited messages to the last port that successfully communicated with that logical RTU (for UDP). For TCP we would use the established connection between the FEP and RTU.

The next problem was how do we want to define our communication contexts which also had serious implications for event management. This issue also got into another problem which was how do we define our hot-stand by Ethernet polling.

While initially the thought wasn't too appealing, allowing the hot-standby Ethernet Master station to have the same DNP address for each of the primary/standby processes was the most logical decision in our case. They have separate IP addresses and will use the same port numbers. Only one of these Master will communicate at a time. When the Master switches/fails over, the new Master will - first, send a reset link to each device and then perform integrity scans.

The advantages of this approach were that one logical RTU would be configured to communicate with either the primary or backup Master. This simplified data reporting and event buffer management. Once events were reported and a Master acknowledged that message, those events could be removed from the buffer. If the acknowledge was not received or lost, then those events would be ready for reporting to the new Master when it came on line. The only exposure (If you can call it that) was if an unsolicited data packet was sent to the primary master and then that Master failed before processing the data. However, that data would be reported when the other Master came on line and performed an integrity scan (hopefully the slave device would be smart enough to know that it sent the data and not continue trying to send the unsolicited data packet to the first master) minimizing exposure of lost data. On an editorial note, some devices support sending unsolicited data packets forever, care and thought is suggested with these devices (which from a system administrator stand point, I think this is bad form – again the concept of this will probably get you when you least expect it to or least want it too).

There is really not a good definition of communication contexts between servers and clients using DNP. Recently some good work in this area from the DNP Technical Committee has been seen.

Another good outcome of this exercise was the use of TCP to support serial port expansion of our RTUs. This allows a terminal server to be connected to the same LAN as the RTU and communication with devices serially connected to the terminal server.

Other Issues That Warrant Consideration...

Another issue occurs when more than one host needs to access the same IED; either functionally related hosts or independent hosts requiring concurrent access. Now the concern is centered about how the IED manages communications to these hosts; does it treat them as independent sessions (communications contexts) or does it present its data on a first come first gets it basis or, some other approach.

Other issues which need to be dealt with include the use of Dynamic IP (DHCP) assignment and address resolution. Not all networks support (or even allow) static IP assignments for connected devices. There are two issues here, one how do I, at any given time, know the IP of a particular device and, could the IP I do know about for a device change? The first issue is resolved by using DNS and 'named devices' and asking for the IP of the IED by name, the second issue is resolved by asking the DNS server for the IP of a device by name at the beginning of each session (connection attempt).

Finally, there is a special class of IED which acts as a door (gateway) to multiple IEDs that it manages and, this IED is the only way that an outside device can access the IEDs it manages. Note, these 'down stream' IEDs could be real physical devices for which the Gateway device acts as a media converter (e.g. IP to Serial) or, they may be Logical IEDs that the gateway acts for because it is acting as a protocol converter or the available information is simply being partitioned.

Finally There Was Testing...

Among other items, the entire test consisted of performing the following for 100 hours:

- 52 serial communication channels communicating at (nominally) 56,000 baud using DNP protocol. Each serial channel used will had one RTU (each with 408 analog and 408 status points).
- 20, individually addressed by DNP address and unique port numbers for each IP address, Ethernet RTUs shared over 2 IP addresses (each with 408 analog and 408 status points). Test was run using DNP over UDP and TCP.
- Total - 29,376 Analog Points (excluding pseudo points)
- Total - 29,376 Digital Inputs (excluding pseudo points)
- Approximate Total - 1000 Digital Outputs (excluding pseudo points)
- Approximate Total - 500 Analog Outputs (excluding pseudo points)
- Approximate Total - 10 Counters
- 25% of analogs change every 2 seconds
- 10 SOEs every 10 seconds
- The system had 2 sets of hot standby Ethernet masters. Both of the on-line masters poll at the same time.
- Class 1 Data scanned as follows – All RTUs scanned every second using Class 1 polls and checking IIN bits to poll for Class 2 and Class 3 data. All polls will use qualifier x06. Note: in previous testing the Ethernet RTUs took as much as 400 milliseconds to

respond to a poll, about 150 milliseconds to deliver a response set and for the Master to ack 4 packets of data for an average of about 450 – 550 milliseconds for a transaction to occur (start to finish). Note: all data in Ethernet RTUs defined as Class 1 (due to the latency in RTUs responding and the fact that the largest data packet generated consumed 1 mSec of bandwidth so using class polling didn't have any benefits).

We learned a lot from the testing. Including:

1. Yes, it really works!!
2. The Utility must get heavily involved and develop understanding of where they want to go and how to get there with the vendors involved.
3. DNP needs to have some important issues cleared up to simplify this process.
4. We can switch between Ethernet and Serial Communications as backup and primary communication (this was a pleasant surprise).
5. We ran 10 DNP over TCP/IP and 10 DNP over UDP/IP RTUs at the same time and there was no visible performance difference.
6. While running DNP over TCP/IP, we failed between the backup and primary Master by failing the LAN. Failovers were accomplished very quickly and the failover performance was so close to the UDP failover performance that there was no difference. We were very pleased with this result.
7. During the full testing, the 10 MB data acquisition LAN (Using UDP) activity averaged 5% with very few rare spikes to 12%; collisions were less than .02%. We added a packet generator to the data acquisition LAN and found that at 29% LAN activity our 2 second updates went to 5 second updates which is still tolerable and at 55% LAN activity the system performance was marginal.
8. While you don't really think of RTU response latency as mattering in the serial world, it really does in the Ethernet world. We found this to be a key factor in determining how many devices could go on a LAN based on the desired update rates. The issue is not the volume of data but device latencies. This presents new challenges to RTU manufacturers plus the move to Ethernet makes SNMP, telnet, and other items highly desirable but few manufacturers have moved towards this direction (but I think it will not be long). Anyhow in our lab we found that an unloaded RTU had a 50 to 100 millisecond delay before it responded to a request for data. In our testing on the factory floor we found delays ranging from 100 to 500 milliseconds depending on how busy the RTU was (in rare cases even 1 sec if the RTU was very busy and a large amount of data had to be reported). Averages are presented in the table below:

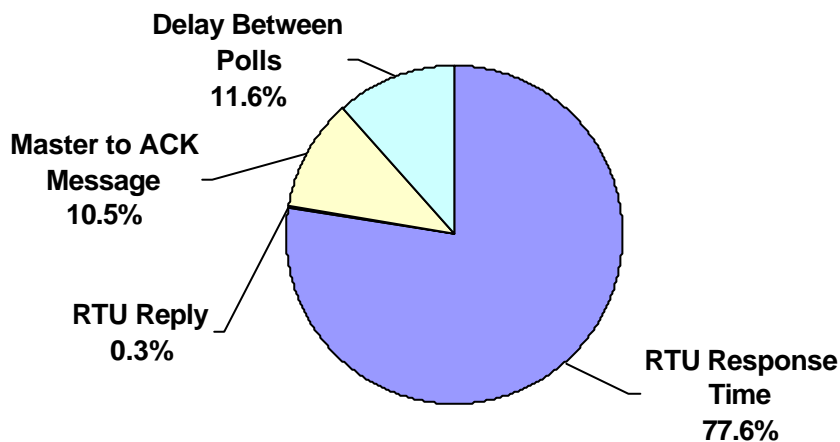
	Ethernet	Ethernet	%BW	%BW	Serial
	Test mSec	Field mSec	Test	Field	in mSec
RTU Rx Delay	133	54	77.6	62.6	349
Rx Length	1	1	0.3	1.2	140
Rx to Ack Delay	18	11	10.5	12.9	19
Total Transaction Time	172	86	11.6	23.4	507
- 20 mSec delay assumed from Ack to complete next Tx					

On the test floor, the average latency was about 130 mSec. In our shop, we observed about 55 mSec for single RTUs that were not loaded . This issue was not anticipated but probably should have been. At this point you could argue that it is not effective to have a 10 MB LAN for RTU communication and one of our solutions under consideration is to go with 4 separate 2.5 MB LANs, other options include a hardware upgrade for the RTUs, other changes to the RTU, other LAN topology changes, etc. We do like the Ethernet environment for data acquisition and the benefits it brings along and feel these far outweigh the performance issue. At this point, using the field data, we would only be able to get updates every 5-6 seconds from each of 60 RTUs. Of course it will take us another 4 years to install that many RTUs and we hope an RTU vendor solution is available by then.

Interesting to note is all the hoopla necessary to get a 1 mSec data packet delivered and confirmed. The data packets here were a little over 1k which is quite a lot for SCADA Data but nothing on a 10 MB LAN. The data was 0.3% of the bandwidth required for the whole transaction!!

Ethernet Bandwidth Utilization Breakdown

Test Data



Some other food for thought...

The use of DNP communications occurs via two primary physical medias; serial devices (where DNP was originally implemented) and the IP world (the topic of discussion). By inserting an appropriate interface between the DNP Applications and the physical media and some configuration information, I can make the actual media used transparent to the DNP application. Now as a system architect/programmer I can concern myself with how best to acquire the information that I am interested in.

However, to achieve this level of abstraction I need to have the underlying transport mechanisms behave in a manner which is consistent with my abstract model. The following deals with serial access, TCP over IP and UDP over IP separately, but I will attempt to illustrate parallels where they exist.

Serial Communications:

In this scenario, I have two serial channels on my computer (referred to as the host from now on) each containing multiple IEDs. Unless my host is acting as a gateway to other computers, only my host can communicate with the devices attached to the serial channels and, only the devices attached to the channel can communicate with other devices attached to the same channel.

NOTE: There is nothing preventing multiple hosts being attached to this channel.

This in fact establishes a communications context (physical). Within the channel's context all devices must have unique addresses so that their messages can be identified from one another. More importantly, each channel can have exactly the same addresses because each channel forms an exclusive communications context. One minor problem exists here, since my host can communicate with the two channels, each of which contains IEDs with the same addresses; how do I present, to other applications, data obtained from two separate devices on separate channels, but both having the same address? There are probably as many different ways to solve this as there is sand grains on the beach, but a much simpler approach is to just give all devices a different address which my host communicates with.

Problem solved, eh? What do you do if the owner of one of the IEDs/devices is unwilling to change because it messes up the software in his host?

When a channel contains devices capable of spontaneously (usually referred to as unsolicited data) sending data two things occur: First, the channel and the protocol must support collision avoidance and retries/receipt confirmation so that data can actually get delivered with some certainty. Secondly, we are experiencing a change in communications context. The device now has an independent notion of context, it can be as simple as knowing only about the host or, it broadcasts its message for all other devices to hear and use appropriately or even more complex, in that it can send this unsolicited data to a specific address(s). Usually, the device's communications context is a subset of the Host's communications context but not always.

The use of the broadcast address (above) works well for all devices attached to the serial channel. But this broadcast message does not appear on the other serial channel (out of context) unless of course the host acts as a repeater (re-broadcasting) of the message on the other channel. Of course this brings up a whole lot of other issues regarding broadcasts, repeating, who is allowed, where is it to go, etc.

Let's suppose for the moment that one of the devices attached to my channel is in fact another host (not related to my system) and both of us share access to all the other devices on the channel. Depending on the arrangements between the two hosts, they may share the same communications context, have overlapping contexts (in the sense that they both access a

common set of devices and they each access devices that the other does not) or, they do not share any context at all (just the physical communications media). When they share a communications context, the operation and capabilities of the 'shared' device become crucial, specifically retrieval of consumable event type data (read once and it is gone type).

If a shared device does not support multiple hosts then that device can not be shared, one host or the other may access it, but not both at the same time. More importantly, one of the hosts has to be designated responsible for acquiring the event data and the other must only listen to acquire it or obtain the data from the other host via another means.

If a shared device does support multiple hosts, then the next issue is does it maintain/support individual host contexts(hosts with different addresses)? If it does not, then an agreed access procedure must be implemented by the hosts so that data is acquired by the appropriate host.

If it does, then does the device retain consumable event data for each host independent of the other? If not, then use the previous solution, otherwise we are all happy campers.

Generally speaking, the applications resident on my computer (SCADA) take advantage of the communications lag associated with sending a request and obtaining a response on a given channel by implementing a concept of concurrent parallel scanning of the all available channels. This serves to increase overall responsiveness of the system (detection of field changes) and make effective use of the CPU time that is available. An argument can be made that use of spontaneous reporting of field changes is the most optimum. However, not all changes available are reported via this mechanism for a variety of reasons; communications speeds, number of devices, rate of change (number/quantity) of field data, etc. Consequently, a balance can be achieved using a combination of both the spontaneous and the request/response mechanisms.

IP Communications:

Everything that I described for serial channel based communications is applicable to IP based communications with the exception of one very important thing, there is no longer a physical separation (isolation) of devices. And the devices are now a member of a much larger channel (network) whose usage is no longer restricted to DNP devices and the DNP protocol.

By applying the concept of communications context to the IP world of DNP devices, I can apply my existing host application model for serial channels to the devices in the network. First however, we have to recognize that the network world (where DNP would be used) is substantially larger and more complicated than perhaps the original vision (DNP specification) accounted for. This has come about as a result of innovations by various implementers, understanding of the capabilities and advantages made available by the interconnected nature of the network/internet. Advances in network security technologies has allowed for secure communications from inside a private network to either public internet sites/equipment or other private networks. Also, the demand for access to information by various enterprise components has led to creative/innovative approaches to the dissemination of data available from 'networked' devices. Cost also plays an important role in that smaller organizations find the cost associated

with private networks prohibitive, whereas public networks, with suitable security, are better suited to their needs.

There are a number of outside influences which need to be accounted for when using network communications; the capabilities of the equipment which interconnects various network segments, especially their restrictions, the network environment that the devices are to work in and how the network is managed. I will be the first to admit that my perception is perhaps limited and my approach is possibly unique.

In general, the only issues we have with the current standard or concepts under discussion for TCP are:

- the use of unique DNP address should only be an issue within a communication context or where multiple contexts overlap.
- the ability of a device to initiate an 'unsolicited' connection while I have an existing connection in place with it should be disallowed. Further more the device should know that the connection to an applicable host is there and send the data via the existing connection.
- if devices record consumable event data, then this data must not be discarded if a connection is lost or, a connection does not exist (yet).
- if a device supports multiple hosts, then it must preserve all consumable events data on a per host basis, and independent of the actions/lack of actions by the other hosts.
- there is a real need to remove the 20000 only port restriction for connection initiation. The standard should define this as the default, and that support for additional ports can be provided as required (configurable).

UDP

First, my understanding regarding the use of UDP over TCP. UDP is simpler to implement in a device (usually embedded devices). This has no bearing on the host since most modern OS used today have the ability to use both TCP and UDP and as such are not an issue. UDP avoids the 'connection' overhead associated with TCP and, packet flow management in particular. This is really not an issue since DNP has the ability to control overall flow through the use of Application or even Link Layer confirms. More importantly, once the connection is established there is no real performance difference. This aspect of TCP only becomes an issue if connections are constantly being created and then terminated. UDP has a limited message (packet) size which is enforced by various network equipment that TCP does not. This is only an issue if you try to send larger packets that can traverse the overall network path or the receiving device cannot handle the size of packet you are sending. TCP provides a certain level of guaranteed packet delivery which UDP does not. Again this is dependent on the dynamics of the network you are trying to traverse. The term connectionless and connection oriented have been used, but these refer to the characteristics of the protocol used rather than the use the applications want. In all worth while cases it takes at least two to communicate. If you have two devices involved in communications, you have a 'virtual connection' regardless of the protocol used to transfer messages. Both parties understand the exchange being made and more often than not, it usually involves two way exchanges of information.

Assume for the moment, that the UDP devices send their responses to the IP and port number from which I sent the request. Since we are using UDP, I do not have the benefit TCP provides in that TCP automatically separates incoming packets into streams which can then be associated with a specific program. Rather, I have to create a virtual stream by having the host program associate itself with a particular port which the outstation device sends its replies back to. Now I have retained a communications model across all three transport media with little overall impact to the host. However, if devices always send their responses to port 20000, I must now do two things: create an intermediate process which takes outgoing messages and sends them to the devices IP and port 20000 from my IP port 20000 and accepts incoming messages on port 20000 and distributes them to the process(s) responsible for that device. Secondly, I have to create a management layer which associates a process with a specific outstation DNP and IP address and the DNP addresses MUST be unique across the whole hosts communications context, it no longer becomes an optional convenience for the host and its applications.

Dealing with spontaneous data using UDP should be no different than if I am using a serial port or TCP connection (TCP operation per previous discussion). If a multiport device can direct spontaneous data to a serial port I am listening on, or to the TCP IP and port connection that my host established, why should I not expect the UDP message to be directed to my IP and port? The argument goes "Because with TCP, there is a negotiation which establishes the IP and Port numbers at both ends.". My argument is that as soon as the device receives the UDP message from me, it knows where I am (Source IP and Port number) - no negotiation required. OK, what about where communications are lost? If my host has successfully delivered one message to the device, it knows where to send any subsequent spontaneous messages. If application or even link layer confirmations are used, then the need for a Heart beat message becomes redundant. What about the case where you have not communicated to the device yet? In this case, the device sends a message to the 'default' IP and port. If the host is not there or it is not ready to communicate, it simply ignores the message. It is the responsibility of the outstation to not discard data which has not been successfully transferred (this concept also applies to the case where the host has suddenly stopped communicating). When the host is ready it will initiate communications, and now the outstation knows where to send any subsequent data to.

What about the case where there is no host, simply device to device communications? If the two devices elect to use 20000 as the port, then that would be apparent in the first message by examining the message source fields to determine where to respond.

Use of the DNP broadcast address is applicable only to UDP, and that capability is usually restricted to the sub-net within which the 'broadcaster' exists. As a rule, UDP broadcasts do not traverse segments (sub-nets), intervening network equipment will block them. So its use within a larger network is very limited.

In general, the only issues we have with the current standard or concepts under discussion for UDP are:

- the use of unique DNP address should only be an issue within a communication context or where multiple contexts overlap.

- that a device send spontaneous data to the last known IP and port number for a particular host.
- if devices record consumable event data, then this data must not be discarded if it can not successfully transfer the data to a target host.
- if a device supports multiple hosts, then it must preserve all consumable events data on a per host basis, and independent of the actions/lack of actions by the other hosts.
- removal/loosening of the 20000 port restriction in both directions. In particular, have the outstation respond to the requesters IP and Port or send spontaneous data to the last known IP and Port for the desired host.

Oh, I almost forgot. From the host perspective, I need to be able to communicate effectively with whatever devices the client has. This doesn't mean I like it or that I don't have preferences and that I won't make recommendations, if asked, about the approach that should be used or what equipment best suits the environment I find myself in. However, I am a strong proponent of standards and their use. I must always play nice in the network Neighborhood, otherwise the people responsible for the network, its maintenance, security and configuration can and will make my life miserable.

Bios:

Joe Orth

Education

May 1986, Received Bachelor of Science in Electrical Engineering from Tulane University

September 1994, Received Master of Sciences in Electrical Engineering (Power Option) from the Georgia Institute of Technology

Employment History

Various positions at Tacoma Power since 1991 in Generation, Power Management and Transmission and Distribution Sections. Since 1997, Special Project Manager for Power System Automation & Information Technology Group (staff of 13) that is responsible for capital projects to install a new SCADA/EMS, new substation RTUs, plan and implement 10 MB SCADA WAN, substation and distribution automation projects, and the operation and maintenance of the existing SCADA System. Approximately \$18 million have been budgeted for SCADA related projects from 1996 through the year 2001 with an additional \$10 million estimated through 2006

Elected Chairman of the DNP User Group February 2001.

Previous employers include Southern California Edison Company, City of Riverside P.U.D. and General Signal

David Eastcott

Education

June 1975, graduated with Honors in Communications Engineering Technology from the Southern Alberta Institute of Technology (SAIT)

July 1987, obtained professional designation of Certified Engineering Technologist (C.E.T.)

Employment History

Various positions at Bow Networks Inc. since 1989. Responsibilities included the design, development of SCADA system components, IED/RTU firmware packages and, embedded system development and deployment. Currently hold the position of Senior Project Manager, responsible for the development and implementation of new communications technologies for both embedded systems and more traditional server/workstation systems.

Previous employers include:

EDO Canada Ltd.; Manager of Engineering and responsible for the development and deployment of Multi-processor distributed Navigation systems utilizing embedded technologies for both commercial and military clients,

JMR Instruments Inc.; responsible for development of commercial embedded marine sensor packages,

Willowglen; responsible for development of embedded SCADA components.

Fig. 1 – Tacoma Power PASS Architecture

