DNP3 Overview

Provided by:
Triangle MicroWorks, Inc.
Raleigh, North Carolina
Phone +1 919-870-5101 • Fax +1 919-870-6692
www.TriangleMicroWorks.com

1 History

DNP was originally created by Westronic, Inc. (now GE Harris) in 1990. In 1993, the "DNP 3.0 Basic 4" protocol specification document set was released into the public domain. Ownership of the protocol was given over to the newly formed DNP Users Group in October of that year. Since that time, the protocol has gained worldwide acceptance, including the formation of Users Group Chapters in China, Latin America, and Australia.

In January 1995, the DNP Users Group Technical Committee was formed to review enhancements and to recommend them for approval to the general Users Group. One of the most important tasks of this body was to publish the "DNP Subset Definitions" document, which establishes standards for scaled-up or scaled-down implementations of DNP3.

DNP3 is an open, intelligent, robust, and efficient modern SCADA protocol. It can

- request and respond with multiple data types in single messages,
- segment messages into multiple frames to ensure excellent error detection and recovery,
- include only changed data in response messages,
- assign priorities to data items and request data items periodically based on their priority,
- respond without request (unsolicited),
- support time synchronization and a standard time format,
- allow multiple masters and peer-to-peer operations,
- and allow user definable objects including file transfer.

In 1994, the IEEE Power Engineering Society's Data Acquisition, Monitoring and Control Subcommittee formed a Task Force to review the communication protocols being used between Intelligent Electronic Devices (IEDs) and Remote Terminal Units (RTUs) in substations.

The IEEE Task Force found a very confusing, constantly changing environment that was increasing the cost and time to completion of substation SCADA systems. The IEEE Task Force collected information on approximately 140 protocols and compared them to a list of communication protocol requirements.

This comparison resulted in a short list of protocols that met most of the requirements. This short list was balloted and two serial SCADA protocols tied for being the most acceptable: IEC 60870-5-101 and DNP3.

DNP3 was being supported by an active users group and was being implemented by an increasing number of vendors. IEC 60870-5-101 was being implemented by an ever increasing number of European vendors.

Both protocols had similar characteristics and strengths. The results of these efforts resulted in IEEE Standard 1379-1997 "Trial Use Recommended Standard for Data Communication Between Intelligent Electronic Devices and Remote Terminal Units in Substations".

The recommendations proved to be well received with an ever increasing acceptance. Based on these actions, the standard was updated in 1999 with a new ballot and became IEEE Standard 1379-200 "Recommended Practice for Data Communications Between Remote Terminal Units and Intelligent Electronic Devices in Substations".

2 Layered Architecture

DNP3 is a layered protocol. However, rather than adhering to the OSI (Open System Interconnection) 7 layer protocol, DNP3 adheres to a simplified 3 layer standard proposed by the IEC (International Electrotechnical Commission) for more basic implementations. IEC calls this the Enhanced Performance Architecture, or EPA. (However, DNP3 enhances EPA by adding a fourth layer, a pseudo-transport layer that allows for message segmentation.)

2.1 Physical Layer

The physical layer is primarily concerned with the physical media over which the protocol is being communicated. For example, it handles state of the media (clear or busy), and synchronization across the media (starting and stopping). Most commonly, DNP is specified over a simple serial physical layer such as RS-232 or RS-485 using physical media such as copper, fiber, radio or satellite. More recent applications have implemented DNP3 over an Ethernet connection.

2.2 Data Link Layer

The data link layer manages the *logical* link between sender and receiver of information and it improves the physical channel error characteristics. For, DNP3 this is accomplished by beginning each data link frame with a data link header, and inserting a 16-bit CRC every 16 bytes of the frame. A **frame** is a portion of a complete message communicated over the physical layer. The maximum size of a data link frame is 256 bytes. Each frame has a 16-bit source address and a 16-bit destination address, which may be a broadcast address (0xffff). The address information, along with a 16-bit start code, the frame length, and a data link control byte is contained in the 10-byte data link header.

The data link control byte indicates the purpose of the data link frame, and status of the logical link. Possible data link control byte values include: ACK, NACK, link needs reset, link is reset, request data link confirm (ACK) of frame, request link status, and link status reply. When a **data link confirmation** is requested, the receiver must respond with an ACK data link frame if the frame is received and passes CRC checks. If a data link confirmation is not requested, no data link response is required.

2.3 Pseudo-Transport Layer

The pseudo-transport layer segments application layer messages into multiple data link frames. For each frame, it inserts a single byte function code that indicates if the data link frame is the first frame of the message, the last frame of a message, or both (for single frame messages). The function code also includes a rolling frame sequence number which increments with each frame and allows the receiving transport layer to detect dropped frames.

2.4 Application Layer

The application layer responds to complete messages received (and passed up from the transport layer), and builds messages based on the need for or the availability of user data. Once messages are built, they are passed down to the pseudo-transport layer where they are segmented and passed to the data link layer and eventually communicated over the physical layer. The total length of received messages is indicated by pseudo-transport layer as it appends data link layer frames, each with their own indicated length.

When the data to be transmitted is too large for a single application layer message, multiple application layer messages may be built and transmitted sequentially. However, each message is an independent application layer message; their only association with each other is an indication in all but the last message that more messages follow. Because of this possible fragmentation of application data, each application layer message is referred to as a **fragment**, and a message may either be a **single-fragment message** or a **multi-fragment message**.

Application layer fragments from Master DNP3 stations are typically **requests** for operations on data objects, and application layer fragments from Slave DNP3 stations are typically **responses** to those requests. A Slave DNP3 station may also transmit a message without a request (an **unsolicited response**).

As in the data link layer, application layer fragments may be sent with a request for a confirmation. An **application layer confirmation** indicates that a message has not only been received, but also been parsed without error. (On the other hand, a data link layer confirmation, or ACK, indicates only that the data link frame has been received and that it passes CRC error checks.)

Each application layer fragment begins with an **application layer header** followed by one or more object header/object data combinations. The application layer header contains an application control code and an application function code. The **application control code** contains an indication if the fragment is one of a multi-fragment message, contains an indication if an application layer confirmation is requested for the fragment, contains an indication if the fragment was unsolicited, and contains a rolling application layer sequence number. The application layer sequence number allows the receiving application layer to detect fragments that are out of sequence, or dropped fragments.

The application layer header function code indicates the purpose, or requested operation, of the message. While DNP3 allows multiple data types in a single message, it only allows a single requested operation on the data types within the message. Example **function codes** include: Confirm (for application layer confirmations), read and write, select and operate (for select-before-operate, or SBO, controls), direct operate (for operation of controls without SBO), freeze and clear (for counters), restart (both cold and warm), enable and disable unsolicited messages, and assign class (discussed below). The application layer header function code applies to all object headers, and therefore all data within the message fragment.

3 Database Organization

In DNP3, data is organized into data types. Each data type is an **object group**, including:

- binary inputs (single-bit read-only values),
- binary outputs (single-bit values whose status may be read, or that may be pulsed or latched directly or through SBO type operations),
- analog inputs (multiple-bit read-only values),
- analog outputs (multiple-bit values whose status may be read, or that may be controlled directly or through SBO type operations),
- counters,
- time and date,
- file transfer objects,
- and others.

For each object group, or data type, one or more data points exists. A **data point** is a single data value of the type specified by its object group.

Also within each object group, object group variations exist. An **object group variation** is typically used to indicate a different method of specifying data within the object group. For example, variations of analog inputs allow for transfer of the data as 16-bit signed integer values, 32-bit signed integer values, or as 32-bit floating point values.

As described above, an application layer message may contain multiple object headers. An **object header** specifies an object group, a variation of the object group, and a range of points within that object group variation. Some application layer header function codes indicate that object data follows each object header; other function codes indicate that there is no object data in the message – instead multiple object headers, if present, follow each other contiguously. For example, a read request message fragment only contains object headers that describe the object groups, variations, and point ranges that are requested to be read and responded; a read response message fragment contains object headers and the request object data.

DNP3 allows **object point ranges** to be specified in a variety of ways. For request messages, object points ranges may consist of

- a request for all points of the specified object group,
- a request for a contiguous range of points beginning with a specified starting point and ending with a specified stopping point,
- a request for a maximum quantity of points,
- or with a list of requested points.

For response messages, object point ranges typically consist of either a contiguous range of points beginning with a specified starting point and ending with a specified stopping point, or with a list of points. For response object point ranges that consist of a list of points, a point number precedes each data object. The number of points in the list is specified as part of the object point range.

4 Reporting Model

Many of the object groups have corresponding, but separate, object groups that contain **change data**. Change data represents only points that have changed for a specifically corresponding object group. For example, object group number 1 represents binary inputs (considered **static data**), and object group number 2 represent binary input change data. When a point in object group 1 is detected to have changed, a change event in object group 2 for the same point number is created. Including only points that have changed in response messages allows smaller, efficient messages. Such reporting schemes are called **report-by-exception**, or **RBE**.

For each change data point, a time can be associated with the change; each detection of a data value that changes is considered a **change event**. At any given time, it is possible to have multiple change events for some points, and no change events for other points.

In DNP3, object groups, and the data points within them, can be further organized into classes. This provides an efficient method of requesting data; a simple (and small) message can be sent to request all data in a specific class (referred to as **scanning for class data**). There are four classes defined in DNP3. Class 0 represents all static (not change event data). Classes 1, 2, and 3, represent different priorities of change event data. By associating different change event data with different classes, the classes can be requested with varying periodic rates.

Assuming class 1 contains the highest priority change event data and class 3 contains the lowest priority change event data, a class 1 poll would ideally be performed as often as possible, a class 2 poll would be performed less often, and a class 3 poll would be performed even less often. For each class data response, only the class data that has changed will be returned – keeping the response messages small and efficient. Finally, to acquire data not associated with either class 1, 2, or 3, an **integrity poll**, consisting of a class 0 scan, would be performed. Because of the possibly large amount of data that will be returned in a class 0 scan, it may not be terribly efficient and should be performed as least often as possible.

End of Document