**Beyond Modbus: Designing SCADA with Other Open SCADA Protocols**

Jacob Brodsky, PE[1]

[1] Jacobs, National Security Solutions 6716 Alexander Bell Drive Columbia, MD 21046
(correspondence: Jacob.Brodsky@jacobs.com, Phone: 443-285-3514)

## Introduction

The Modbus protocol is among the oldest communications protocols still in use today. Designed in the late 1970s, it has been ported to many platforms and has been specialized for various industries. Today, many continue to use it. Its primary advantage is simplicity. The software required to run it is very lightweight. There are several open source protocol stacks available for Modbus.

However, Modbus is showing signs of age. Because it pre-dates the practice of management via a standards committee, there are many variants of the Modbus protocol. Each variant introduces new opportunities for compatibility headaches. There is no support for more complex data types such as floating point numbers, long integers, strings, and the like.

Another issue with Modbus is scalability. It is a real time protocol. For it to capture transient events, a master must poll an RTU at least twice as fast as the fastest transient event (Nyquist Theorem). The more data one has to get from the RTU, the more traffic there will be. The more resolution required for the data, the more frequently one must poll.

## Data Types

There are other real time protocols. Some, such as the ODVA's EthernetIP, have been around for 20 years and offer stronger data typing, and a producer-subscriber architecture. Another option is the MQTT protocol from the "Internet of Things" crowd. It is a similar concept, except with significantly more marketing behind it and the foolish notion that it should have anything to do with the Internet.

These protocols can solve many problems with data types. In Modbus, there are only two basic types of data: Binary and 16 bit registers. Sending a 32 bit counter can be messy. Is the high order word in the low address or is the high order word in the high address? This is known as Endian-ness. There is no consistent application of Endianness among the variants of Modbus.

Furthermore, if there is string data, there is no standard for how it should be sent. Some send a word with the string length first and then the string going upward in registers. Some use a null terminator for the string. Some send the string going downward in registers. The Modbus standard doesn't discuss strings at all. How it is mapped in to the registers is entirely a choice for the user. Note that ModbusIDA is trying to clean up some of these practices, however they have only been doing this since 2004 and the installed base of other Modbus variants is very large.

Nevertheless, the common thread behind these other protocols is that most are "real time" protocols. In other words, a master asks what the values are and the remote replies with the values. Issues of telecommunications propagation time, cryptographic overhead latency, and the like are ignored.

The issue of various latencies is not new. In fact, more than 30 years ago, SCADA vendors found a work-around: Event based reporting.

## Real-Time versus Event based Reporting

In a real-time system, the master asks what each value reads right now. It has to enumerate each and every thing it wants to know about and they have to be reported each and every time. When it gets those values, it then has to compare them against its dead-band database.

Contrast that with event based reporting where the master asks "what's new?" The remote in an event based system will respond only with those things that have changed enough to merit reporting. The remote's configuration is where those decisions get made. In a well tuned system, the typical response from the remote is "nothing new." The dead-bands are in the remote, not the master.

The difference between these models is stark. Consider a door switch on a cabinet containing an RTU. If you want to know what the status of that door switch is, you have to poll it every few seconds, or someone might quickly open and close the door and not get detected.

However, with an event oriented protocol, the door switch will trigger an alert no matter how short a period of time the door was open. It isn't necessary to poll the unit frequently. There will be an event, even if it isn't reported right away.

## Reporting Events

There are four ways to create an event at a remote device.

1. Value
2. Time
3. Integrity
4. "On Demand"

These methods are elaborated below.

### Events on Value

In the case of value triggered events, the value has to change past some amount that would be "interesting" enough to merit reporting. These are a few of the more common algorithms for deciding if an analog value is worth reporting:

a. Relative
b. Absolute
c. Integrated difference
d. Swinging Door
e. Some combination of the above

The first is relative value. If the reading changes from the last reported value by more than some amount, report it. This is one of the simplest, commonly used methods.

The second is absolute value. As the value goes greater or less than some regular value on the scale, it makes a report as to what it is. Between those reporting value limits, it does not report. Generally, there is a settling delay after the value crosses over that boundary and then the actual value is reported. This method is often used where overall resolution of a point isn't all that critical, but large changes are.

A third method is to monitor the difference between the current value and the last reported value. That difference is integrated over time. When it exceeds some threshold, it reports the value in an event. This guarantees that small changes over time aren't covered up.

Another method is known as the "swinging door" algorithm. As a value' s rate of increase or decrease changes, the change of rate triggers a report. This will place a reading at every corner where the rate changes.

These are just some of the dead-band methods that are commonly in use today. It is also possible to combine these methods on a single point. However, the complexity of this sort of dead-bandit must coincide with the operational needs.

Many operators are used to seeing "grass" on their trend displays and will object if the dead-band algorithm removes it. They may express doubts of data integrity because that "liveliness" of the data is somehow missing. To address those concerns, it is often useful to present the operators with a counter of the number of messages received from each RTU. As long as they see that counter incrementing, it means that they are capable of getting data. This concern is actually an education problem. The difficulty is that real-time reporting frequently does display this kind of "noise" in the data, and it is just that: noise. Reporting noise is a waste of bandwidth.

### Events on Time

Next, there are triggers on time. With an event oriented system, the remote is usually expected to know what the time of day is. At some time, either a regular interval or some delay after another event, the remote can be configured to take a snapshot what a value is. That snapshot gets reported as an event. This method is often used for counters. Counter applications include tipping bucket rain gauges, pulse counters for power meters, and the like.

These time triggers can also be used for launching internal calculations, such as volumetric flow in or out of a storage tank. The internally calculated results then become another event. These locally calculated results can have much greater resolution than the results calculated at a SCADA master.

### Events on Integrity

A point is more than just a value or a state. There are usually auxiliary "flags" or integrity information on the point. For example, the remote may be receiving I/O from another networked instrument. If a breaker trips, that instrument may go offline, though the remote is still capable of reporting. So it is legitimate to make an event out of the "panel not available" state. Another point state may relate to a self-integrity condition, such as a loop power supply alarm; or it may be because of an indication that the point is being forced to a specific value or state locally (usually for testing purposes).

Binary Status information can also have triggers. There are triggers on state, on flags, and on time. The flags may include features such as a "chatter filter." Chatter filters typically use some algorithm to reduce the number of events when a state goes back and forth rapidly. For example, a door switch may be going on and off because people are going in and out of a building. The only thing one needs to know is that someone did open the door at some point within a given time or reporting period. The number of times it does so in between polling periods is irrelevant. So a door switch may have a "chatter filter" to prevent the remote from reporting every opening and closing of the door.

### On-Demand Events

An operator might want to know "right now" what a reading is. For example, just before starting a pump, someone may want to know how much energy that run used. So they might want to read the pulse counter right then from the electric meter, not seven or eight minutes later. They would then start the pump. After it runs for some period of time and shuts down, they may trigger an additional counter reading. This could be used to make better estimates of how much power a particular run cycle uses.

## Thin versus Thick Master stations

Doing all this processing in the remote station makes things easier for the master station. All of the event oriented processing outlined above can either take place in the master or it can take place in the remote. If it takes place in the remote, the master doesn't have much to do except put a time stamp on when the message was received, maintain a buffer for query by the rest of the SCADA system, and service requests against that buffered state information. If the dead-band computation takes place in the master, the load on the telecommunications infrastructure is higher because the remote must report more traffic. This places higher reliability demands on the telecommunications infrastructure. It also affects resiliency issues

The key difference is that by pushing the processing to the field, there is no need to rely upon the telecommunications being available. The data resolution and time stamping is much better, and local calculations such as flow totalization can be more accurate.

If the telecommunications between the RTU and Master fail, events will continue to be generated. When communications are restored, those events can be made available to the SCADA system.

Unfortunately, this brings up another concern: Many SCADA systems use OPC drivers, specifically OPC-DA. This is a problem for event oriented protocols because the OPC-DA API designed only for real-time protocols. There is no provision for a remotely time-stamped event. So if the remote event has a timestamp on it, that timestamp is often difficult to capture in the Historian database. While there are ways of capturing the timestamp, many historians do not make efforts to capture those remote timestamps on an event.

The problem with the latter scenario is that one can have a communications outage of several hours. When the communications problem is resolved, the old events will come flooding back in to the SCADA system. There may be a significant number of alarms, and these would be incorrectly assigned to the current time, not when they actually occurred.

Most OPC-DA drivers of event oriented protocols make the local event timestamp available. Unfortunately, because there is no standard for how the local timestamp should be represented, it is unlikely that the historian will capture that data.

## Multiple Master Considerations

One advantage of real-time protocols is that they do not have significant state machines behind them. There is no reason why several other masters couldn't poll the same remote. However, because it is a real time protocol, each master may get a slightly different answer. This usually isn't operationally significant, but it may confuse analysts attempting to understand the slight differences between masters reading the same change of custody meter.

The issue with some event oriented protocols is more significant. The best practice is to maintain a separate remote instance for each master. This can be more difficult to set up and maintain. The problem with event oriented remotes is that the current protocols require them to keep track of which master has what events. Once an event is read and acknowledged, it is purged from the event buffer. If more than one master polls the same remote instance, they would "steal" each other's events, and neither of them would have a complete picture of what was going on.

There is development underway for a less stateful version of an event oriented protocol that pushes event tracking back to the Master station. This way, multiple masters can poll the same remote instance without "stealing" each other's events.

## Is it Worth Changing Protocols?

Entry level SCADA systems are usually quite primitive. There is no need for the complexity of an event oriented protocol. In a water utility operation, if a typical water/waste-water SCADA system has fewer than 100 remotes, it usually isn't worth the effort to make the change.

There is overhead and extensive management that has to take place with an event oriented protocol. It's not for those who are just looking to read a couple gauges in the field. However, event oriented protocols are more resilient and more easily secured.

Nevertheless, the author's personal experience of more than 30 years at a large water utility was that over any given five year period, the point count approximately doubles. During that time the number of remotes will have increased by about 20%. In other words, the number of remotes increases slowly, while the number of points increases more rapidly. This means each remote gets "fatter" over time (the number of points per remote increases significantly).

An event-oriented protocol can handle this scale-up better than a real-time protocol. So for medium to large water utilities, it may be worth investigating event oriented protocols such as IEC 60870-5-101, IEC 60870-5-104 (known as the '101 and '104 protocols) or DNP3 (commonly used in North America).

As a rule of thumb, for water and waste-water SCADA applications, if all of the RTUs have point counts of less than 20 status points and four analog inputs, it's not worth using event oriented features. The savings of event orientation is not worth the complexity and overhead. However, if there are any RTUs in the system with more than 50 points, it's worth a look.

Water and Waste-water SCADA systems tend to be pretty sparing with the I/O that is defined. This may well change in the coming years as needs for tracking energy usage increase and as variable frequency drive use becomes more commonplace.

## Summary

Event orientation is more difficult to manage because the dead-bands are in the field, not the master. They do have some overhead, so using it on an RTU with few points may not be worthwhile. However, it does offer increased data resolution, it does scale up better, and it is more resilient in poor telecommunications environments.

**ABOUT THE AUTHOR**

**Jacob Brodsky, PE:** After 31 years doing everything ICS and SCADA from the Instrumentation to the Historians and servers at the Washington Suburban Sanitary Commission; Ten Years on the DNP3 Technical Committee, Co-Authoring and Co-Editing two editions of a Handbook on SCADA/Control Systems Security by CRC Press, and co-founding the SCADASEC e-mail list; it is possible that Jake might know something about SCADA security and design. Jake is a Professional Engineer of Control Systems, registered in the State of Maryland; and a Senior ICS Security Engineer for the National Security Solutions of Jacobs