WHITE PAPER

New Technology Automates Arduous Modbus Routing Setup in Gateways

Dr. Sean Chen
Product Manager

MOXA®

New Technology Automates Arduous Modbus Routing Setup in Gateways

Abstract

When a lot Modbus devices need to be monitored and controlled, engineers usually have to spend a lot of time planning the topology of these devices and segmenting them into different subgroups. Moreover, they are also weighed down by the mundane task of keying in hundreds of Modbus slave IDs to set up each Modbus gateway's Modbus slave ID routing table. Thus, engineers are constantly on the lookout for solutions that address this pain point when setting up multiple Modbus devices. This white paper takes a closer look at different routing technologies and their pros and cons, as well as a new technology that helps save time and costs when configuring and managing a large number of Modbus devices.

Introduction

For many applications, embracing the Industrial Internet of Things (IIoT) has paid huge dividends. One noticeable trend is the migration of a large number of serial devices to Ethernet-based networks, allowing plant managers to tap the full potential of their legacy devices by unlocking previously unused data. However, adding value to these serial devices comes at a cost in terms of time and effort, especially when dealing with a large-scale Modbus network. For example, let's take a look at how the complicated nature of this type of network presents itself in building automation where hundreds to thousands of serial-based temperature controllers communicate via Modbus RTU protocol. These temperature controllers need to be controlled and monitored in a control room, which uses Modbus TCP. At this point, the non-interoperability of protocols becomes an issue. A tried-and-tested solution to overcome non-interoperable protocols is installing high-port-density Modbus gateways that convert serial to Ethernet as well Modbus RTU to Modbus TCP, and vice versa. However, engineers still have to figure out how many gateways need to be installed and how many serial ports are needed on each gateway. Therefore, planning a network's topology that involves a large number of Modbus devices to achieve full-fledged connectivity can really test engineers' mettle.

Reality bites

To engineers, spending too much time and effort on planning a Modbus network's topology is counter-productive. For example, they find it especially time-consuming to set up a Modbus slave ID routing table, which lists the connections of Modbus devices (Modbus slave IDs) to specific serial ports on a gateway. Adding to engineers' frustration is a high possibility that things might not go according to plan in the field. Connectivity errors at field sites can undo all

Released on May 26, 2017

the meticulous planning in the office within moments; thus, sending engineers back to the drawing table and redoubling their efforts. A crucial aspect of planning a Modbus network's topology is to eliminate these connectivity errors when dispatching a large number of Modbus requests to the serial devices that are connected to a Modbus gateway. Life would be so much easier for engineers if they didn't have to worry about which serial devices were connected to which serial ports on a Modbus gateway. In an ideal situation, they would be able to just send out Modbus requests to a Modbus gateway, and the latter would automatically find the correct serial port that connects with the target Modbus device. This would iron out many pain points, even when adding new Modbus devices to a system or connecting existing devices to a different serial port.

The Key Challenges

Serial-based devices' response times are generally slower than those of Ethernet-based devices. Their slow response time is even more evident when they are connected to a gateway in a daisy-chain topology, as the one-request-one-response nature of a Modbus protocol leads to a longer polling time. In these types of setups, a one-port Modbus gateway delivers better performance because a supervisory control and data acquisition (SCADA) system can communicate independently with each gateway; thus, shortening the communication gap between the large number of Modbus devices and a SCADA system. However, the management of multiple Modbus gateways is very complicated. Hence, multiport Modbus gateways are more adept at managing a large number of Modbus devices. For example, one 16-port Modbus gateway can replace 16 one-port Modbus gateways. In space-limited applications, it's a win-win situation that frees up physical space and only requires one power cable and one Ethernet cable. In addition, the large number of IP addresses needed for 16 one-port Modbus gateways can be consolidated into a single IP address. For SCADA systems, another benefit is lower connection fees as they are normally charged according to the number of connections.

But multiport gateways are not exactly a breeze when it comes to the management of multiple Modbus devices. Engineers first need to segment all the devices into groups and then connect them to a specific port on the gateway. This is why a well-created Modbus slave ID routing table of a gateway's serial ports is so important, but creating an efficient routing table is time-consuming.

Dispatching a large number of Modbus requests

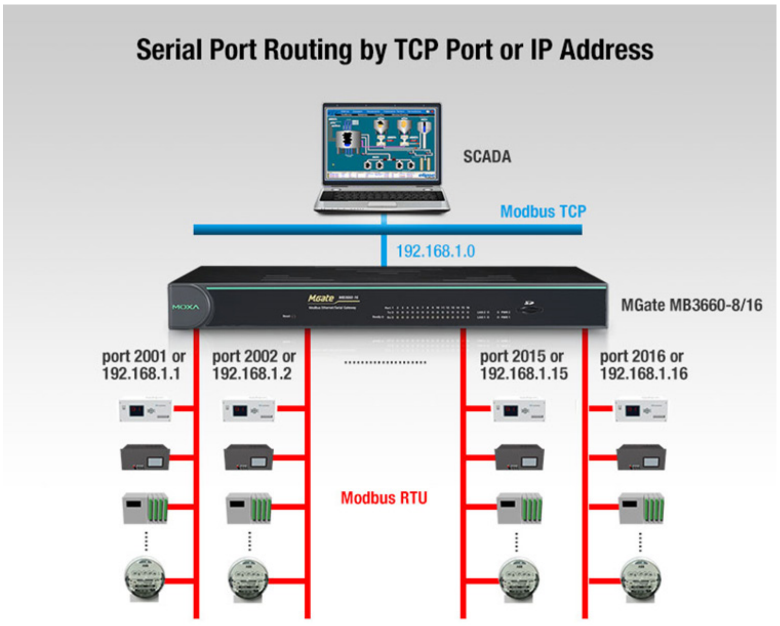
Unlike Ethernet switches, where routing is accomplished automatically through an ARP table, the routing mechanism for Modbus gateways with multiple ports is much more intricate. Currently, two types of routing mechanisms address the different requirements in Modbus-based networks.

Routing by an IP address or TCP port

Some Modbus gateways perform the serial-port mapping functionality via an IP address or TCP port. This mechanism is suitable for engineers who want to monitor field devices in segments. All the Modbus slave devices that are connected to the same serial port through daisy-chain wiring correspond with a specific IP address or TCP port. That is, each serial port on a gateway corresponds with a unique IP address or TCP port. Furthermore, a high-port-density gateway can be used instead of a large number of low-port-density gateways. As previously mentioned, this reduces cabling significantly.

A drawback is that engineers have to manually configure as many IP or TCP connections as the number of serial ports available. In large-scale Modbus environments, systems usually adopt a large number of multiport Modbus gateways, making configuration a time-consuming task—not to mention the extremely high connection fees involved.

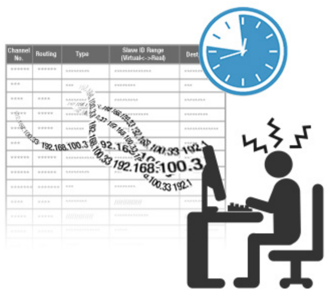
Furthermore, some configuration needs to be done for the Modbus gateways themselves. For example, according to a SCADA system's functionality, a gateway's serial port 1 (as shown in the illustration below) should be set with the IP address 192.168.1.1, serial port 2 with the IP address 192.168.1.2, and so on. With regard to TCP ports, the pattern is similar: set serial port 1 with TCP port 2001, serial port 2 with TCP port 2002, and so on. In other words, a gateway needs to accommodate multiple connections. If there are 16 serial ports on one gateway, then the gateway has to accept at least 16 TCP connections simultaneously.



Furthermore, serial-port mapping with IP addresses or TCP ports needs to be manually maintained by an engineer, which requires the engineer to identify the serial port that a device is connected to, and the corresponding IP address or TCP port.

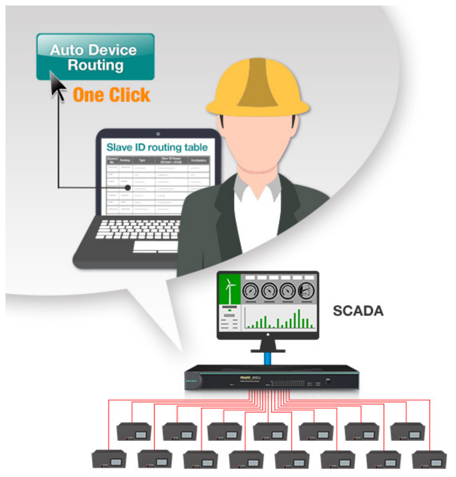
Routing by using a gateway's Modbus-ID routing table

For engineers who care about connection fees and don't need to monitor devices in segments, a more popular option is using a Modbus slave ID routing table. The main purpose of a Modbus slave ID routing table is to indicate which Modbus device (Modbus ID) is connected to which serial port on a gateway. Once a gateway receives a Modbus request for a specific Modbus device, it can dispatch this request via the referring Modbus slave ID routing table to the serial port that connects to the target Modbus device. A SCADA system benefits by using only one IP address or TCP port to communicate with all the Modbus devices that are connected to a gateway, easing the management of Modbus devices and reducing connection fees considerably.



The Modbus slave ID routing table needs to be maintained for troubleshooting and maintenance; however, creating as well as managing a Modbus slave ID routing table is laborious. Also, it needs to be stressed that when engineers come in contact with a Modbus gateway for the first time, it would be as if they are climbing a mountain as they would be completely unfamiliar with routing table settings. They have to bundle the Modbus slave IDs into groups and then connect each group to a different serial port. For example, slave IDs 1 to 5 are connected to serial port 1, slave IDs 6 to 10 are connected to serial port 2, and so on. According to this method, they will have to set the routing rules 16 times for a 16-port Modbus gateway. The situation becomes a lot more taxing if serial port 1, for example, is connected to ID 1, ID 6 and ID 11, and serial port 2 is connected to ID 2, ID 7, ID 12, and so on, because they need to keep track of what IDs are connected to each port. Engineers then will have to set routing rules as many times as the number of Modbus slave IDs. It does not get any easier for engineers when they have to reset the Modbus slave ID routing table for newly added Modbus devices to a system.

Just one click



A new leading-edge technology that automatically detects the Modbus requests from a SCADA system and sets up the Modbus slave ID routing table comes as the answer to engineers' prayers. The patent-pending Auto-Device Routing function only requires a single click to help the gateway detect which serial port is connected to a target Modbus device, allowing it to automatically dispatch a Modbus request to the correct serial port. It automatically creates the routing table, saving significant time and costs as engineers no longer need to manually create the Modbus slave ID routing table anymore, eliminating possible human error in the process. Furthermore, it eliminates the effort needed to double-check the actual connections at field sites. There is no need to refer to a historical Modbus slave ID routing table when adding or removing Modbus devices, saving time and effort.

Conclusion

By automatically creating a routing table, the Auto-Device-Routing technology makes the configuration and maintenance of a gateway's Modbus slave routing table a thing of the past. This patent-pending function features in the Moxa's MGate MB3000 Series, which consists of high-performance Modbus gateways with 2,4,8, or 16 serial ports. The MGate MB3000 Series also supports routing by IP address or TCP port. For more information about the MGate MB3000 Series, please visit: http://www.moxa.com/Event/ethernet-gateways/modbus-auto-device-routing/index.htm