

SCADA

| Contents: | Page no: |
|---------------------------------------|-----------------|
| Introduction | 03 |
| DNP3 Over IP and Protocol Adaption | 04-05 |
| Tunneling Lagency of SCADA | 05-06 |
| SCADA Protocol Translation | 07 |
| SCADA Transport over LLANS with MAP-T | 08-09 |
| Generic web based protocol | 09 |
| Conclusion | 10 |

Introduction:

In the world of networking technologies and protocols, IoT is relatively new. Combined with the fact that IP is the de facto standard for computer networking in general, older protocols that connected sensors and actuators have evolved and adapted themselves to utilize IP. A prime example of this evolution is supervisory control and data acquisition (SCADA). Designed decades ago, SCADA is an automation control system that was initially implemented without IP over serial links, before being adapted to Ethernet and IPv4. A Little Background on SCADA For many years, vertical industries have developed communication protocols that fit their specific requirements. Many of them were defined and implemented when the most common networking technologies were serial link-based, such as RS-232 and RS-485. This led to SCADA networking protocols, which were well structured compared to the protocols described in the previous section, running directly over serial physical and data link layers. At a high level, SCADA systems collect sensor data and telemetry from remote devices, while also providing the ability to control them. Used in today's networks, SCADA systems allow global, real-time, data-driven decisions to be made about how to improve business processes. SCADA networks can be found across various industries, but SCADA is found mainly concentrated in the utilities and manufacturing/industrial verticals. Within these specific industries, SCADA commonly uses certain protocols for communications between devices and applications. For example, Modbus and its variants are industrial protocols used to monitor and program remote devices via a master/slave relationship. Modbus is also found in building management, transportation, and energy applications. The DNP3 and International Electrotechnical Commission (IEC) 60870-5-101 protocols are found mainly in the utilities industry, along with DLMS/COSEM and ANSI C12 for advanced meter reading (AMR).

Adapting SCADA for IP

In the 1990s, the rapid adoption of Ethernet networks in the industrial world drove the evolution of SCADA application layer protocols. For example, the IEC adopted the Open System Interconnection (OSI) layer model to define its protocol framework. Other protocol user groups also slightly modified their protocols to run over an IP infrastructure. Benefits of this move to Ethernet and IP include the ability to leverage existing equipment and standards while integrating seamlessly the SCADA subnetworks to the corporate WAN infrastructures. To further facilitate the support of legacy industrial protocols over IP networks, protocol specifications were updated and published, documenting the use of IP for each protocol. This included assigning TCP/UDP port numbers to the protocols, such as the following:

- ♣ DNP3 (adopted by IEEE 1815-2012) specifies the use of TCP or UDP on port 20000 for transporting DNP3 messages over IP.

- ♣ The Modbus messaging service utilizes TCP port 502.

- ♣ IEC 60870-5-104 is the evolution of IEC 60870-5-101 serial for running over Ethernet and IPv4 using port 2404.

- ♣ DLMS User Association specified a communication profile based on TCP/IP in the DLMS/COSEM Green Book (Edition 5 or higher), or in the IEC 62056-53 and IEC 62056-47 standards, allowing data exchange via IP and port 4059. These legacy serial protocols have adapted and evolved to utilize IP and TCP/UDP as both networking

and transport mechanisms. This has allowed utilities and other companies to continue leveraging their investment in equipment and infrastructure, supporting these legacy protocols with modern IP networks. Let's dig deeper into how these legacy serial protocols have evolved to use IP by looking specifically at DNP3 as a representative use case. Like many of the other SCADA protocols, DNP3 is based on a master/slave relationship. The term master in this case refers to what is typically a powerful computer located in the control center of a utility, and a slave is a remote device with computing resources found in a location such as a substation. DNP3 refers to slaves specifically as outstations. Outstations monitor and collect data from devices that indicate their state, such as whether a circuit breaker is on or off, and take measurements, including voltage, current, temperature, and so on. This data is then transmitted to the master when it is requested, or events and alarms can be sent in an asynchronous manner. The master also issues control commands, such as to start a motor or reset a circuit breaker, and logs the incoming data. The IEEE 1815-2012 specification describes how the DNP3 protocol implementation must be adapted to run either over TCP (recommended) or UDP. This specification defines connection management between the DNP3 protocol and the IP layers, as shown in Figure 3.12. Connection management links the DNP3 layers with the IP layers in addition to the configuration parameters and methods necessary for implementing the network connection. The IP layers appear transparent to the DNP3 layers as each piece of the protocol stack in one station logically communicates with the respective part in the other. This means that the DNP3 endpoints or devices are not aware of the underlying IP transport that is occurring.

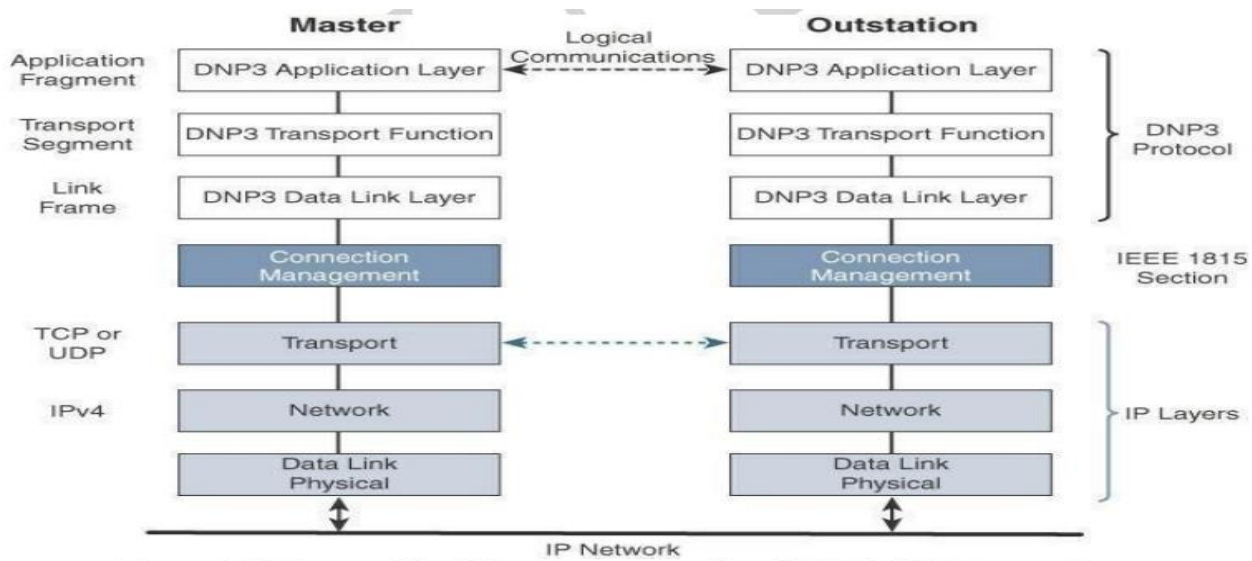


Figure 3.12 Protocol Stack for Transporting Serial DNP3 SCADA over IP

In Figure 3.12, the master side initiates connections by performing a TCP active open. The outstation listens for a connection request by performing a TCP passive open. Dual endpoint is defined as a process that can both listen for connection requests and perform an active open on the channel if required. Master stations may parse multiple DNP3 data link layer frames from a single UDP datagram, while DNP3 data link layer frames cannot span multiple UDP datagrams. Single or multiple connections to the master may get established while a TCP keepalive timer monitors the status of the connection. Keepalive messages are implemented as DNP3 data link layer status requests. If a response is not received to a keepalive message, the connection is deemed broken, and the appropriate action is taken.

3.7.2.2 Tunneling Legacy SCADA over IP Networks

Deployments of legacy industrial protocols, such as DNP3 and other SCADA protocols, in modern IP networks call for flexibility when integrating several generations of devices or operations that are tied to various releases and versions of application servers. Native support for IP can vary and may require different solutions. Ideally, end-to-end native IP support is preferred, using a solution like IEEE 1815-2012 in the case of DNP3. Otherwise, transport of the original serial protocol over IP can be achieved either by tunneling using raw sockets over TCP or UDP or by installing an intermediate device that performs protocol translation between the serial protocol version and its IP implementation. A raw socket connection simply denotes that the serial data is being packaged directly into a TCP or UDP transport. A socket in this instance is a standard application programming interface (API) composed of an IP address and a TCP or UDP port that is used to access network devices over an IP network. More modern industrial application servers may support this capability, while older versions typically require another device or piece of software to handle the transition from pure serial data to serial over IP using a raw socket. Figure 3.13 details raw socket scenarios for a legacy SCADA server trying to communicate with remote serial devices.

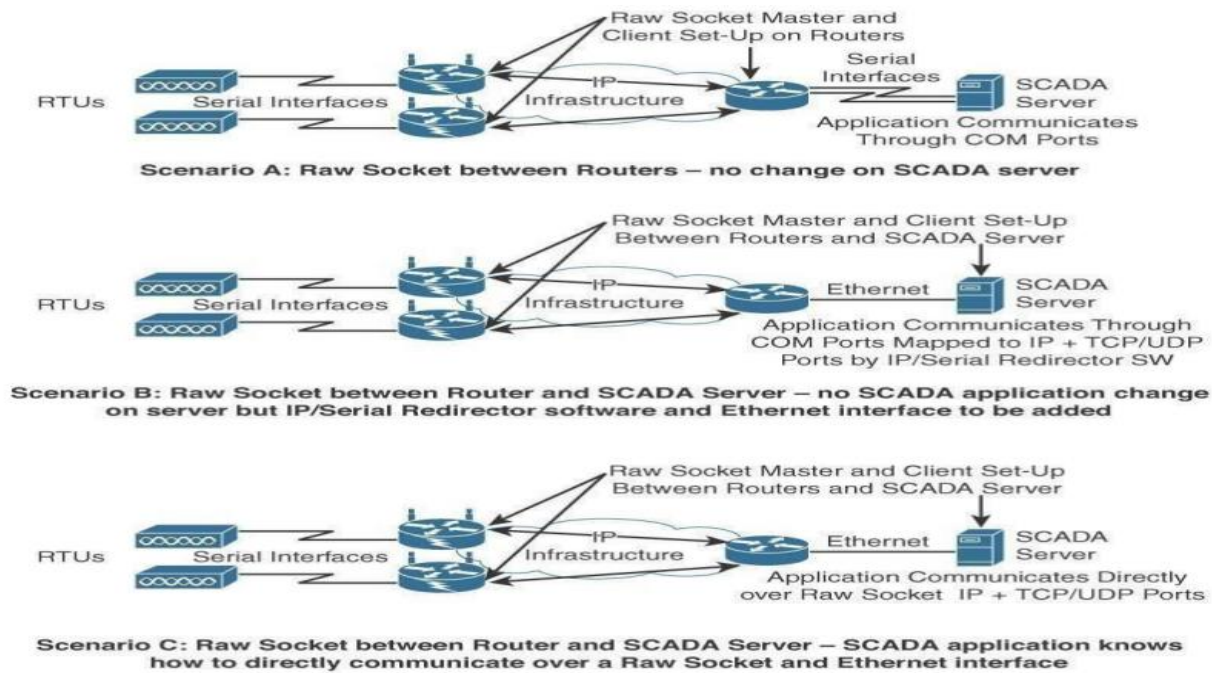


Figure 3.13 Raw Socket TCP or UDP Scenarios for Legacy Industrial Serial Protocols

In all the scenarios in Figure 3.13, notice that routers connect via serial interfaces to the remote terminal units (RTUs), which are often associated with SCADA networks. An RTU is a multipurpose device used to monitor and control various systems, applications, and devices managing automation. From the master/slave perspective, the RTUs are the slaves. Opposite the RTUs in each Figure 3.13 scenario is a SCADA server, or master, that varies its connection type. In reality, other legacy industrial application servers could be shown here as well. Scenario A in Figure 3.13, both the SCADA server and the RTUs have a direct serial connection to their respective routers. The routers terminate the serial connections at both ends of the link and use raw socket encapsulation to transport the serial payload over the IP network. Scenario B has a small change on the SCADA server side. A piece of software is installed on the SCADA server that maps the serial COM ports to IP ports. This software is commonly referred to as an IP/serial redirector. The IP/serial redirector in essence terminates the serial connection of the SCADA server and converts it to a TCP/IP port using a raw socket connection. Scenario C in Figure 3.13, the SCADA server supports native raw socket capability. Unlike in Scenarios A and B, where a router or IP/serial redirector software has to map the SCADA server's serial ports to IP ports, in Scenario C the SCADA server has full IP support for raw socket connections.

3.7.2.3 SCADA Protocol Translation

As mentioned earlier, an alternative to a raw socket connection for transporting legacy serial data across an IP network is protocol translation. With protocol translation, the legacy serial protocol is translated to a corresponding IP version. For example, Figure 3.14 shows two serially connected DNP3 RTUs and two master applications supporting DNP3 over IP that control and pull data from the RTUs. The IoT gateway in this figure performs a protocol translation function that enables communication between the RTUs and servers, despite the fact that a serial connection is present on one side and an IP connection is used on the other.

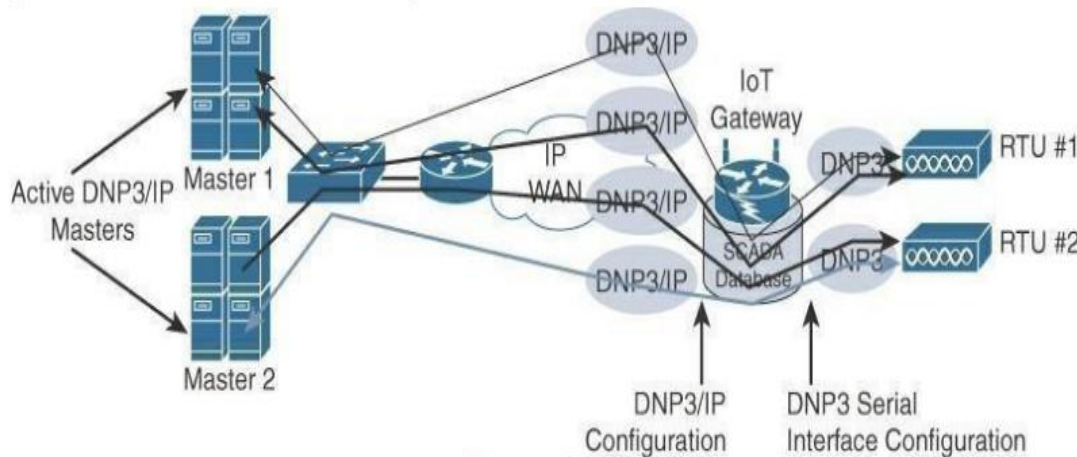


Figure 3.14 *DNP3 Protocol Translation*

By running protocol translation, the IoT gateway connected to the RTUs in Figure 3.14 is implementing a computing function close to the edge of the network. Adding computing functions close to the edge helps scale distributed intelligence in IoT networks. This can be accomplished by offering computing resources on IoT gateways or routers, as shown in this protocol translation example. Alternatively, this can also be performed directly on a node connecting multiple sensors. In either case, this is referred to as fog computing.

. 3.7.2.4 SCADA Transport over LLNs with MAP-T

Due to the constrained nature of LLNs, the implementation of industrial protocols should at a minimum be done over UDP. This in turn requires that both the application servers and devices support and implement UDP. While the long-term evolution of SCADA and other legacy industrial protocols is to natively support IPv6, it must be highlighted that most, if not all, of the industrial devices supporting IP today support IPv4 only. When deployed over LLN subnetworks that are IPv6 only, a transition mechanism, such as MAP-T (Mapping of Address and Port using Translation, RFC 7599), needs to be implemented. This allows the deployment to take advantage of native IPv6 transport transparently to the application and devices. Figure 3.15 depicts a scenario in which a legacy endpoint is connected across an LLN running 6LoWPAN to an IP-capable SCADA server. The legacy endpoint could be running various industrial and SCADA protocols, including DNP3/IP, Modbus/TCP, or IEC 60870-5-104. In this scenario, the legacy devices and the SCADA server support only IPv4 (typical in the industry today). However, IPv6 (with 6LoWPAN and RPL) is being used for connectivity to the endpoint. 6LoWPAN is a standardized protocol designed for constrained networks, but it only supports IPv6. In this situation, the end devices, the endpoints, and the SCADA server support only IPv4, but the network in the middle supports only IPv6.

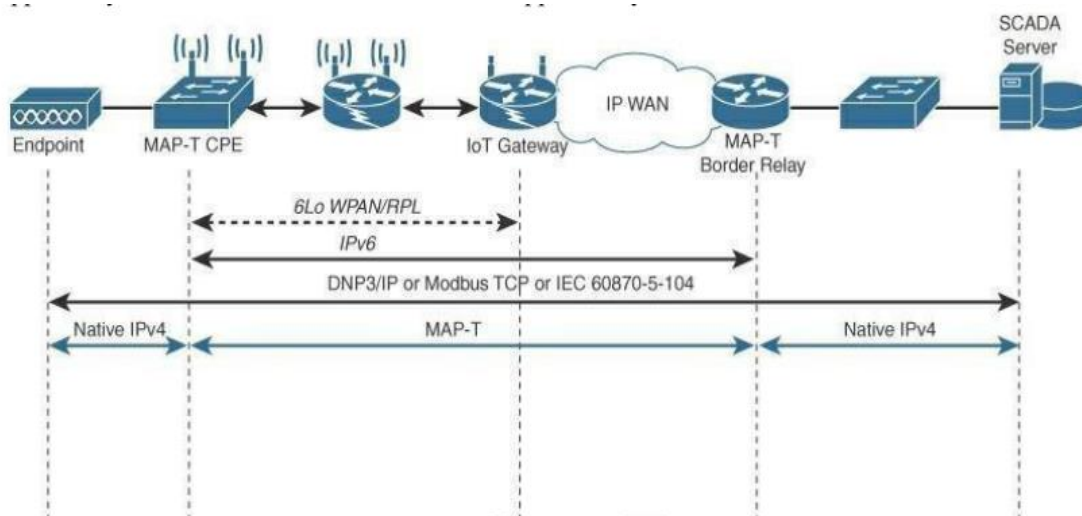


Figure 3.15 DNP3 Protocol over 6LoWPAN Networks with MAP-T

The solution to this problem is to use the protocol known as MAP-T. MAP-T makes the appropriate mappings between IPv4 and the IPv6 protocols. This allows legacy IPv4 traffic to be forwarded across IPv6 networks. In other words, older devices and protocols can continue running IPv4 even though the network is requiring IPv6. In Figure 3.15 the IPv4 endpoint on the left side is connected to a Customer Premise Equipment (CPE) device. The MAP-T CPE device has an IPv6 connection to the RPL mesh. On the right side, a SCADA server with native IPv4 support connects to a MAP-T border gateway. The MAP-T CPE device and MAP-T border gateway are thus responsible for the MAP-T conversion from IPv4 to IPv6. Legacy implementations of SCADA and other industrial protocols are still widely deployed across many industries. While legacy SCADA has evolved from older serial connections to support IP, still it can be expected to see mixed deployments for many years. To address this challenge, OT networks require mechanisms such as raw sockets and protocol translation to

transport legacy versions over modern IP networks. Even when the legacy devices have IPv4 capability, the constrained portions of the network often require IPv6, not IPv4. In these cases, a MAP-T solution can be put in place to enable IPv4 data to be carried across an IPv6 network.

3.7.3 Generic Web-Based Protocols

Over the years, web-based protocols have become common in consumer and enterprise applications and services. Therefore, it makes sense to try to leverage these protocols when developing IoT applications, services, and devices in order to ease the integration of data and devices from prototyping to production. The level of familiarity with generic web-based protocols is high. Therefore, programmers with basic web programming skills can work on IoT applications, and this may lead to innovative ways to deliver and handle real-time IoT data. For example, an IoT device generating an event can have the result of launching a video capture, while at the same time a notification is sent to a collaboration tool, such as a Cisco Spark room. This notification allows technicians and engineers to immediately start working on this alert. In addition to a generally high level of familiarity with web-based protocols, scaling methods for web environments are also well understood—and this is crucial when developing consumer applications for potentially large numbers of IoT devices. Once again, the definition of constrained nodes and networks must be analyzed to select the most appropriate protocol. On non-constrained networks, such as Ethernet, Wi-Fi, or 3G/4G cellular, where bandwidth is not perceived as a potential issue, data payloads based on a verbose data model representation, including XML or JavaScript Object Notation (JSON), can be transported over HTTP/HTTPS or WebSocket. This allows implementers to develop their IoT applications in contexts similar to web applications. The HTTP/HTTPS client/server model serves as the foundation for the World Wide Web. Recent evolutions of embedded web server software with advanced features are now implemented with very little memory (in the range of tens of kilobytes in some cases). This enables the use of embedded web services software on some constrained devices. When considering web services implementation on an IoT device, the choice between supporting the client or server side of the connection must be carefully weighed. IoT devices that only push data to an application (for example, an Ethernet- or Wi-Fi-based weather station reporting data to a weather map application or a Wi-Fi-enabled body weight scale that sends data to a health application) may need to implement web services on the client side. The HTTP client side only initiates connections and does not accept incoming ones. On the other hand, some IoT devices, such as a video surveillance camera, may have web services implemented on the server side. However, because these devices often have limited resources, the number of incoming connections must be kept low. In addition, advanced development in data modeling should be considered as a way to shift the workload from devices to clients, including web browsers on PCs, mobile phones, tablets, and cloud applications. Interactions between real-time communication tools powering collaborative applications, such as voice and video, instant messaging, chat rooms, and IoT devices, are also emerging. This is driving the need for simpler communication systems between people and IoT devices. One protocol that addresses this need is Extensible Messaging and Presence Protocol (XMPP)

Coclusion

The evolution of SCADA systems from serial-based to IP-based networks has significantly enhanced industrial automation. With the integration of Ethernet, IoT, and web-based protocols, SCADA now offers real-time monitoring, remote access, and improved scalability.

The adoption of protocol translation, tunneling, and MAP-T solutions ensures that legacy SCADA systems can function within modern network infrastructures, including IPv6-based LLNs. Additionally, web technologies like HTTP, MQTT, and WebSockets simplify data exchange and cloud integration.

Looking forward, AI, edge computing, and enhanced cybersecurity will further drive SCADA innovation, making industrial automation more adaptive, intelligent, and secure. The convergence of SCADA, IoT, and web technologies will enable industries to achieve greater efficiency, automation, and interoperability, shaping the future of smart industrial systems

References:

- Srinivasa, K. G. Internet of Things. CENGAGE Learning India, 2017.
- Arduino Documentation. Arduino Uno Rev3 Hardware. [Online]. Available:
<https://docs.arduino.cc/hardware/uno-rev3/>
- Robu.in. Arduino Pin Configuration. [Online]. Available: <https://robu.in/arduino-pin-configuration/>
- Scribd. Introduction to Arduino - PowerPoint Presentation on Arduino. [Online]. Available: <https://www.scribd.com/presentation/688375786/Introduction-to-Arduino-PowerPoint-Presentation-on-Arduino>