

Progressing towards Ethernet 'Protocol Irrelevance'

By: Lynn August Linse

<http://www.linse.org/kopihaus>

ABSTRACT

One of the true opportunities of Ethernet is its ability to support a large number of protocols at the same time without hardware change. While industrial philosophers bemoan the decades old crusade of "Why can't we all just implement the SAME single protocol ...", vendors (especially smaller ones) have no financial choice but to offer the many Ethernet protocols end users request.

Even today Ethernet I/O products exist which offer concurrent support for Modbus/TCP, Ethernet/IP, and several other protocols. Giving the falling cost of Flash memory, it's conceivable that within the year Ethernet I/O products will support up to a dozen Ethernet-carried protocols concurrently.

So even without a single, industry-wide protocol, this enables the marketing and application of one-model-fits-all type products. This talk outlines some of the TCP/IP issues that enable this organized chaos, some of the data modeling issues required for concurrent support for multiple protocols, and some product design examples that successfully leverage multiple protocols to produce a whole product that is "greater than the sum of its parts".

Dawn of a new era

The ability of Ethernet with TCP/IP to reliably mix many protocols on a single wire to a single device will revolutionize automation in a gentle, evolutionary way. While in the early days it will cause trouble, confusion, and even great gnashing of teeth, as automation implementers gains wider experience this mixing of many protocols will solve it's own problems in a smooth and relatively painless way.

The Single Minded Serial Data Stream

The lack of a reliable way to mix multiple protocols on a single physical wire is one of the many limitations of serial data communications. Subtle differences between packet framing, timing, and the special treatment of character sequences require each protocol to live on its own wire. This limitation is a prime cause for the headaches of evolving multi-vendor systems over time. Multi-vendor only? Even successive generations from a single-vendor often suffer this headache of mixing versions of the same protocol on new and existing media.

For example, consider 2 of the most popular serial protocols today: Allen-Bradley's DF1 and Modicon's Modbus/RTU. Within DF1 the character DLE (binary 16) is special and only appears alone within a control command, otherwise it must be doubled ("escaped"). By contrast, Modbus has no special interpretation of character values and allows a DLE to used anywhere, yet Modbus does place special meaning on short pauses within data transmission. So while Modbus requires a steady, uninterrupted stream of data, DF1 with its well-delimited end-of-message symbol can survive large gaps within the data stream.

Therefore a DF1 device seeing Modbus message *will* misinterpret the data stream, as will a Modbus device seeing DF1 messages. The only partial solution is to create a hybrid device that is both Modbus and DF1-aware and maintains a real-time state machine to manage the message per the current context. But even that would fail at times to properly communicate with other non-hybrid devices.

The closest products come today is to allow users to select (enable) one of many protocols on a single port in an exclusive manner. Each protocol offers some benefit, but hides any benefit of the others. All small controllers offer Modbus as an option – but once enabled, users cannot use the vendors programming and diagnostic tools on-line anymore. The user must manually re-enable the vendor's protocol, reboot the device, or rely upon multiple serial ports and multiple data paths.

Dramatic Difference within TCP/IP

TCP/IP is dramatically different. Besides the much-mentioned IP Address, each data packet includes a port number in the range of 1 to 65,535. If an IP address can be viewed as the telephone number which routes your call to the correct building, then the port number can be viewed as the extension number used to select the exact phone to reach within that building.

Since TCP/IP allows thousands of applications to register interest in data from a single wire, it is trivial for multiple protocols to share that same wire. As each data packet is received, TCP/IP routes it to the correct application within the device. Since each application can expect a different protocol, there is no reason a single device cannot support dozens of protocols at the same time.

So in our previous example, if Modbus data is arriving on TCP port 502 and DF1 data is arriving on TCP port 3001, there is no ambiguity. As long as 2 applications have properly registered their interest in these ports, the underlying TCP/IP network software reliably separates & delivers the data to each application.

Beginning of Media & Protocol Irrelevance

So the wide spread introduction of TCP/IP into industry offers an intriguing new potential - with a single Ethernet interface it is now conceivable to create a slave device (or data server) that can be polled by all major protocols. For example, imagine a flow computer or power meter or controller that can be queried simultaneously by Modbus/TCP, Profibus under IP, Ethernet/IP (CIP or DeviceNet under IP), HSE (Foundation Fieldbus under IP), and many other protocols. Each protocol arrives on a distinct TCP or UDP port number; each data packet is unambiguously delivered to the correct application for handling.

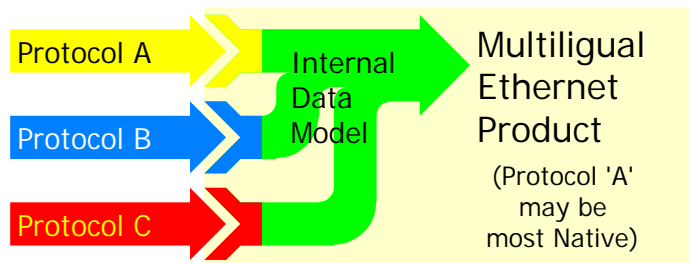


FIGURE 1 – MODERN MULTI-LINGUAL ETHERNET DEVICE

Such multi-lingual slaves/servers don't need to fully implement each protocol – only the portions required for normal, operational reading and writing of real-time data. For example, a flow computer with a legacy proprietary protocol encapsulated within TCP/IP (say protocol "A") can make full use of this protocol for configuration and maintenance. This doesn't hinder the addition of code to allow Modbus, Profibus, and CIP protocols under TCP/IP to also read/write the real-time data of the flow computer. Plus unlike traditional serial communications, TCP/IP allows multiple concurrent accesses and has no inherent limit restricting users to a single host. Thus it is easy to implement a redundant master system since each master can access the slave independently.

Shifting Complexity from Macro to Micro Level

So while this creates a more complex slave/server device (the “micro” level), it greatly simplifies the already complex control system (the “macro” level). Now the main control system can standardize on a protocol like Modbus/TCP or Ethernet/IP for operational data transactions, yet still retain legacy protocols for use with legacy systems or maintenance and troubleshooting tools.

At first glance this seems to promise chaos, yet it is a chaos that will naturally reorder itself over time. This is born out by modern computer history. Over the past 25 years, I feel two “rules” have proven themselves time and again:

- 1) Data communication standards that start life in “committee” have a very small probability of becoming widely used standards in the field. Instead, most successful data communication standards start life as successful ad-hoc or proprietary standards developed by a small group of people, which later enter committee to evolve in a community way.
- 2) When multiple data communication standards offering roughly the same functionality exist, a natural evolution moves users toward the standard offering the “best value” and others will disappear or retreat into niche markets. Technical quality has little to do with the survival or extinction of data communication protocols.

The past history of Ethernet, ISO/OSI, MAP/TOP, Novell, and the common TCP/IP “protocol suite” all bear witness to these rules. ***So I propose that migration through this chaos is the only way a single, industry-wide protocol standard is possible.*** Since the current data communication situation is largely chaotic, the only solution is a design that allows a gradual alignment and reordering of the chaos over time.

Evolution, not Revolution

While the painful birth and death of MAP/TOP back in the 80’s had many factors, one undisputable cause is the great disruption it’s adaptation required to existing systems. It required the division of industrial equipment into 2 groups: new higher priced devices that ***can***, and old (already paid for) devices that ***cannot***. Given the money realities of business and the need to migrate plants over time (preferably without interrupting production!), any Utopian single standard requiring such a complete change is destined to fail.

But the chaos of Ethernet and TCP/IP is destined to succeed. Systems and equipment will progress thru a series of phases:

- 1) We start with an average, chaotic collection of proprietary protocols on various serial and proprietary media.
- 2) As new devices are added, add products offering diverse protocols on TCP/IP over Ethernet.
- 3) Retrofit existing equipment with Device Server® technology (serial-to-Ethernet converters) using legacy serial protocols encapsulated within TCP/IP over Ethernet.
- 4) At this point, users have begun a steady migration to a single network standard. This is ½ of the dream, and since Ethernet and TCP/IP support diverse media including wireless, fiber optics, and high-speed wide-area networking, the benefits of this step alone will pay great dividends.
- 5) As older equipment is replaced, newer Ethernet devices can be installed. But the reason for replacement is no longer for network integration - the lifetime of equipment is greatly extended.
- 6) As industrial standards evolve and solidify under TCP/IP, the Device Servers can be firmware (flash) upgraded to map the changing standards into the older legacy protocols. As multiple “standards” jockey for dominance, devices can be programmed to support them all.
- 7) Then perhaps one glorious day users will wake up and say “Hey, the utopian dream of a single industrial protocol is true – and I have it”. It wasn’t implemented in a day or a single project, but evolved over the years through the normal course of change and repair.