Innovasic®

White Paper

What is PriorityChannel™?

Abstract

PriorityChannel is a combination of hardware and software that passes critical data from an Ethernet-based network to an application regardless of the traffic on the network. Sounds simple enough, but considering all of the traffic an end device must process, things get complicated in a hurry. Relying on an Ethernet switch to take care of things isn't enough, and simple filtering doesn't help much either. If there is traffic on the network that an Ethernet stack must take care of, then the application must wait to get its critical data – unless, of course, the application uses PriorityChannel. This paper explains the issues with getting critical data to an application using Ethernet, and how PriorityChannel overcomes these issues.

Background

Ethernet has become wide-spread in factory automation. Regardless of the flavor of Ethernet (Profinet, EtherNet/IP, etc.) the point is to get time sensitive data to and from field devices so the factory can operate smoothly. Back when the network was a serial fieldbus or even point-to-point 4-20 mA, the network had little to no influence on the operation of the field device. However, with the application of Ethernet in the factory, there is much more going on over the network than simply passing time sensitive data to and from the end field devices.

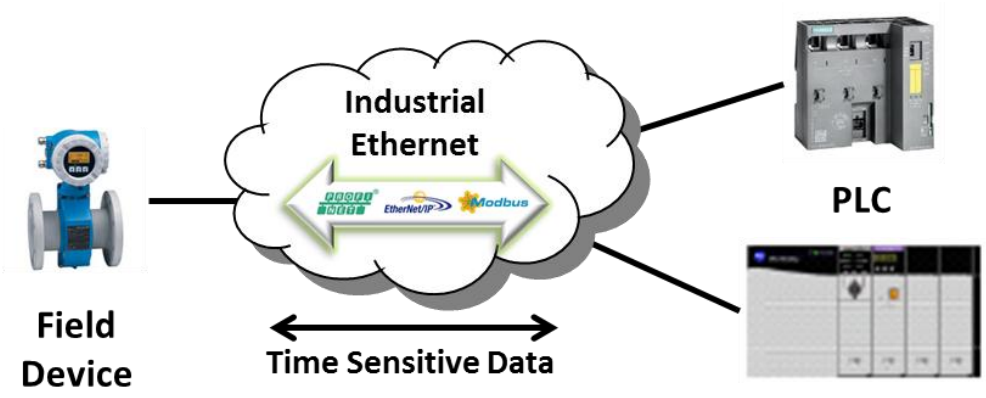
AUTHORS:

Dave Alsup
dave.alsup@innovasic.com

Tom Weingartner
tom.weingartner@innovasic.com

Innovasic, Inc.
5635 Jefferson St. NE, Suite A
Albuquerque, NM 87109 USA
www.innovasic.com

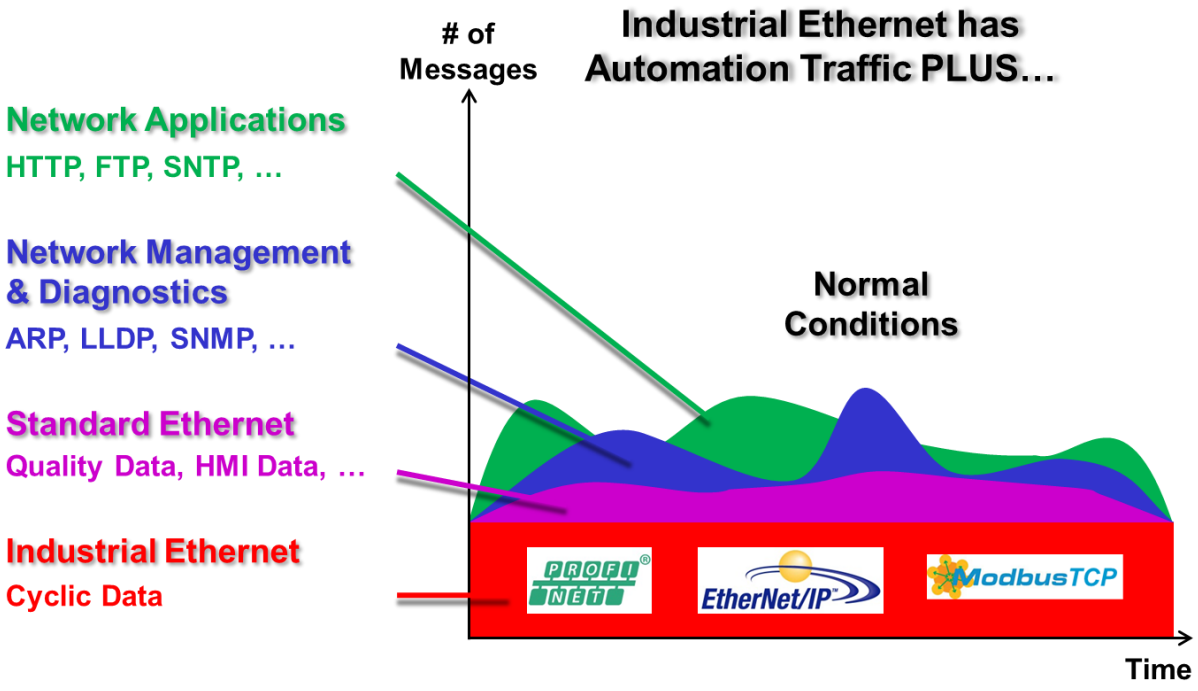
May 14, 2012



All of this activity on the network is a "necessary evil" and can dramatically influence the response of the field device. This activity can be so overwhelming for devices that time sensitive data stops being managed by the device. The following sections describe the different types of network activities and explain how PriorityChannel provides a device with immunity from this activity.

Network Traffic Components

As shown in the figure below, network traffic out on the wire consists of Industrial Ethernet traffic plus traffic from standard Ethernet sources, Network Management protocols, diagnostic tools, and Network Applications. The Industrial Ethernet traffic shown in red is analogous to the traffic out on a traditional field bus system. This is time sensitive traffic commanded by a PLC or DCS, and contains cyclic updates for field device data parameters.



Standard Ethernet traffic is from other non-automation applications out on the wire and in the end device. This can be data destined for a printer from an HMI device on the network, or quality data destined for the front office. Typically, standard Ethernet traffic is minimized on the factory floor.

Network Management and Diagnostic traffic is required to keep the network infrastructure working properly. SNMP is used to query and control elements on the network (i.e. switches, HMIs, Sensors, Actuators, PLCs, printers, etc.). Link Layer Discovery Protocol (LLDP) provides information about the layout of a network and can be used in re-configuring a system, as is the case when a faulty device is replaced. Address Resolution Protocol (ARP) is used by many higher-level applications to query for addressing information. In an improperly organized network, an ARP storm can result from broadcast messages being endlessly propagated through the network, demanding a high percentage of network bandwidth. This situation can happen during commissioning or maintenance simply by inadvertently creating a loop by connecting two ports of an unmanaged switch or misconfiguring a managed switch. Every field device on the network (i.e. sensor, actuator, PLC, HMI, etc.) must handle this additional traffic.

Network Applications traffic is generated by "layer 7" programs and protocols that are used as utilities by other programs. Many field devices have a web server to configure the device, and this uses HTTP. Transferring files to and from a PLC or DCS typically uses FTP. Keeping time on the network would use SNTP or IEEE 1588. Within industrial Ethernet protocols themselves, there is much traffic that is necessary for establishment and maintenance of the connections between PLCs and devices. These include RPC, DCP, as well as TCP and UDP.

Switches and Segmentation Can't Do It All

It is common practice to isolate field devices from non-relevant Ethernet traffic (i.e. traffic not intended for the field device) by segmenting the network with properly configured Ethernet switches. Of course, such a practice assumes only managed switches are used to provide segmentation and switch configuration capability. It also assumes networks use a star topology in order keep traffic from other devices to a minimum. For other topologies such as a line or a ring, the switch resident on the field device should be configured to pass only messages addressed to the field device to the device's application. In other words, a field device's switch should isolate the field device's application from traffic destined for other network devices, such as another field device's real-time traffic, an HMI sending data to a printer, or a PLC sending data to the front office. However, there remain three classes of network traffic that a field device's switch cannot isolate from its application. These are:

- Network management and network application traffic. This is traffic to and from the device other than the time-critical data required by the core application of the device. This can include such things as connection establishment and maintenance messages to/from a PLC, SNMP traffic for network maintenance, LLDP messages for network topology management, FTP or HTTP traffic for other features of the device, etc. For example, if another device wants to access a field device's webserver or if a PLC requests network management information, there is no way for a switch to be configured to ignore these messages.
- Broadcast and Multicast traffic. While a properly configured switch can isolate most undesired broadcast and multicast traffic from a device, all devices must receive and analyze some broadcast messages, primarily to support ARP requests, otherwise basic communication cannot take place. Therefore, it is common to limit the maximum number of broadcast packets that are allowed in a period of time or use other techniques to handle an overload of broadcast traffic. Whichever scheme is used, the field device still needs to process a reasonable percentage of this traffic in order to maintain basic communications.
- Erroneous or malicious traffic. This is traffic directed to the field device, but for which there is no supported function or service in the device. For example, another element on the network can attempt to open a connection to a socket that is not supported by the device. The field device must still service this request, taking time away from the device's real-time communication. When this type of action happens repeatedly in an attempt to hack or crash the device, it can be considered malicious.

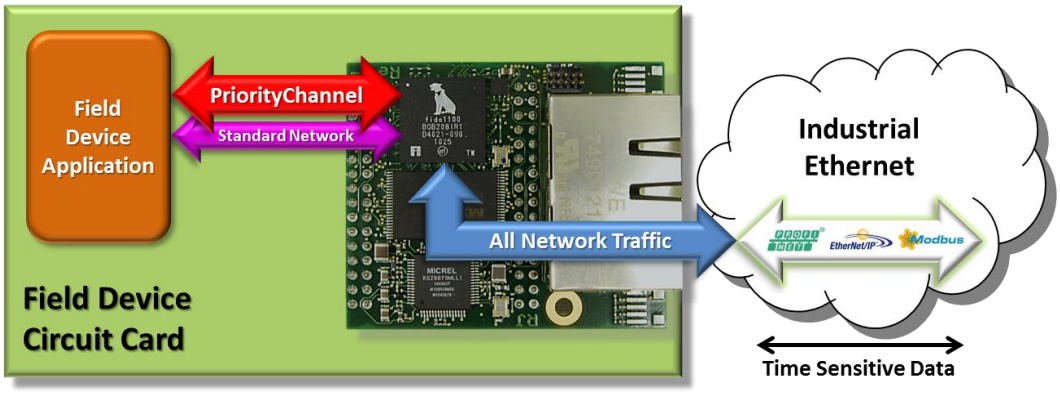
Since a switch cannot isolate a field device's application from the above classes of traffic, the problem is compounded when such traffic spikes to unexpected levels. It is not unusual for many devices on a network to send particular types of maintenance traffic at roughly the same time, for example. This simultaneous transmission results in a large number of packets to be managed in a relatively short time. In other cases, a network with a mix of field devices with different network management capabilities (i.e. Class A and Class B devices in the same PROFINET network) can result in packets going where they don't belong. On a PROFINET network, LLDP packets may be propagated from multiple devices causing a large spike in traffic for a network with many field devices.

The bottom line is that there is a lot more going on over an Ethernet network than on a serial fieldbus network. Segmentation and switches can only help so much, especially as factory networks evolve. Standard techniques for controlling traffic at the Ethernet switch level do not solve the problem of guaranteeing the real-time traffic and processing in the midst of this sort of traffic. This leaves field devices vulnerable to eventual performance problems or even failure over time as network traffic increases. Wouldn't it be better if field devices could inherently protect themselves from having to respond to all of these types of traffic?

Of course the answer is "Yes!", and that is the purpose of PriorityChannel. Rather than throw more MHz or GHz at the problem (which just adds to power consumption, complexity, and cost), the real answer is to prioritize the time sensitive application messages in the field device. There is no need to spend precious processing cycles on non-critical network messages when you don't have to. The next section describes how PriorityChannel lets field devices inherently protect themselves from non-critical traffic.

How PriorityChannel Works

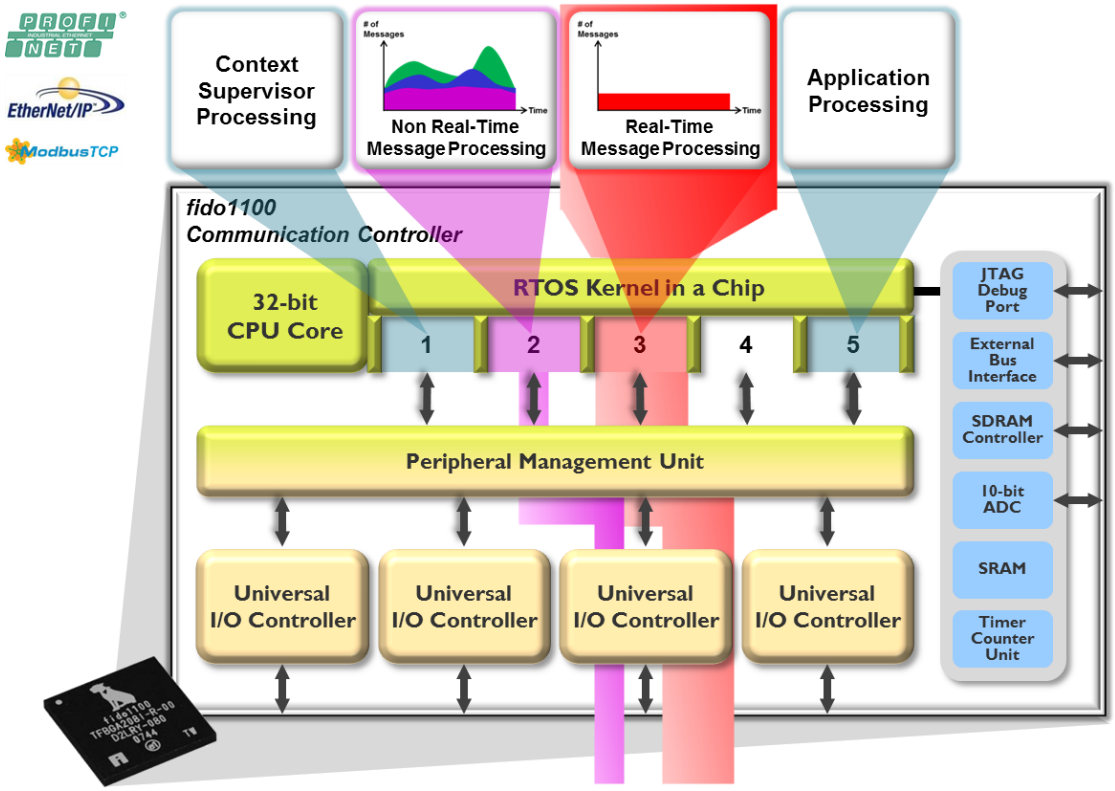
One of the attributes of Ethernet is that it is possible to distinguish different message types. At a top level, it is easy to think of PriorityChannel as finding the important, time sensitive messages and separating them from the standard network data, effectively setting up a "channel" for the time critical data. In addition, this channel has a higher processing "priority" than the standard network data which allows the critical data to pass directly to the field device application uninterrupted. This concept is shown in the figure below.



It is important to point out that PriorityChannel is not just fancy filtering. While certain message types are handled by the switch, most messages are passed to the communication controller – in this case, the Innovasic fido1100. There, the messages are identified so message

processing can take advantage of the fido1100's unique architecture. Let's take a look at how the fido1100 chip architecture gives PriorityChannel superior network performance.

The heart of the fido1100 is a 32-bit CPU that contains five contexts, or "threads", implemented in the silicon itself. These are the same type of contexts that would be implemented as threads in a real-time operating system (RTOS) kernel. All of the scheduling, priorities and memory protection found in an RTOS are implemented directly in silicon in the fido1100. This RTOS-Kernel-in-a-Chip allows separation of communication tasks as shown in the figure below. Real-time message processing can be assigned to one context, while non-real-time message processing can be assigned to another context. Further, and most importantly, these contexts can be given different priorities so the lower priority, non-real-time context, does not interrupt the higher priority, real-time context.



In traditional architectures, the TCP/IP stack is a non-re-entrant thread so it must be processed to completion even if the application has a higher priority message ready for processing. This means an ARP message, for example, will complete its processing by the TCP/IP stack even if it is time for the field device application to get its data to the PLC. If the PLC requires this data at a 2 ms rate, for example, it may arrive at 2 ms one time and 3 ms

another time. In this example, the ARP message blocks the processing of the real-time message even though that is not what is best for the system. One way to mitigate this problem is to use faster and faster processors, but this doesn't completely solve the problem, especially as network traffic grows.

With the fido1100 architecture, all non-real-time processing is allocated to a low priority context and real-time message processing is allocated to a high priority context. When a real-time message needs processing it can immediately interrupt all lower priority contexts. It doesn't matter if the TCP/IP stack is non-re-entrant, the hardware context switching takes precedence. And, because the prioritization and task switching happens in the silicon hardware, not in software, the context switching happens in just one clock cycle. In the case of the ARP message example above, TCP/IP stack processing is immediately suspended and the device application data is sent to the PLC. Also, if the PLC requires data at a 2 ms rate, it gets data at a 2 ms rate. This performance is guaranteed for all network loading situations – even as network traffic grows.

In addition to the fast context switching, another key architectural feature of the fido1100 is that it identifies Ethernet messages quickly and routes them directly to the appropriate context for processing. This function is performed in the Universal I/O Controller (UIC). The UIC is programmed specifically for the protocol being used. For example with PROFINET, the UIC selects as high-priority only those packets with the PROFINET ethertype, the frame type for the current connection and those sent from the Controller for the current connection. For EtherNet/IP, only packets to the appropriate UDP port are forwarded to the PriorityChannel. Other protocols are managed similarly.

Separate high- and low-priority hardware message queues are also provided on the chip so that the high priority packets can be sent or received even if there are a lot of low priority packets already in the queue. The ARP, FTP, HTTP, SNMP, LLDP and other messages discussed above are identified and sent up to the non-real-time message processing context, while the messages containing real-time data are sent up to the real-time message processing context. In an overload situation, where the processor doesn't have the bandwidth to process all messages that are received and requested, the PriorityChannel processing still takes place on time and completely. The lower priority processing is managed to provide the best practical response. The real-time data packets are not dropped because they have dedicated resources available; instead the low priority packets are triaged and processed on a best-effort basis.

Conclusion

PriorityChannel is a combination of patented hardware and software that allows time-critical application data to get to/from the network without interference from non-time-critical traffic. PriorityChannel's performance is achieved by the unique architectural features on the fido1100 controller chip. Separate on-chip contexts with priority management (RTOS Kernel-in-a-Chip), UIC selection of high-priority packets and separate on-chip message queues combine to allow unmatched performance for real-time packets without the need for very high clock speeds.

PriorityChannel performance can be maintained regardless of network traffic loads. The Innovasic White Paper, "Real-Time Performance of Industrial Ethernet in Field Devices" details PriorityChannel's performance in real network test situations. Since PriorityChannel actually solves the network loading problem in Industrial Ethernet systems, there is no need to throw faster, more expensive and power hungry processors at the problem. There is no need to worry that traffic conditions in the future may cause failures in your products and an expensive re-design to an even faster, more expensive processor. Your investment in a certified solution is protected.