EtherNet/IP over IPv6 - Evolution, not Revolution for the World's Leading Industrial Ethernet Variant

Dayin XU
Research Engineer
Rockwell Automation

Yi YU
Sr. Engineer
Rockwell Automation

Paul Brooks
Business Development Manager
Rockwell Automation

Brian Batke
Principle Engineer
Rockwell Automation

Presented at the ODVA
2012 ODVA Industry Conference & 15th Annual Meeting
October 16-18, 2012
Stone Mountain, Georgia, USA

Abstract

In this paper we explore the technical impact of migration to IPv6 for CIP Technologies and its implications for modern EtherNet/IP devices. The paper will make proposals for the inclusion of IPv6 in EtherNet/IP together with IPv4. The paper will explore the architectural implications for end users of EtherNet/IP technologies when operating in a hybrid EtherNet/IP over IPv4 and IPv6 environment. The paper will outline areas where the EtherNet/IP specification must be enhanced and will make proposals for technical implementations of these changes for consideration by the ODVA community. The paper will also highlight the IPv6 development practice by making dual stack IPv4-IPv6 prototypes of EtherNet/IP configuration software, scanner and adapter devices. The paper will explore the steps that a number of companies have gone through in constructing a multi-vendor proof of concept for partial dual stack IPv4-IPv6 implementation of EtherNet/IP. We conclude that the future proof design of EtherNet/IP ensures that migration to IPv6 can be managed incrementally and gradually by our customers, and at their own paces.

Keywords

IPv6, Dual-stack, EtherNet/IP, Hybrid Application, Industrial Automation, IPv6 Transition, Universal Socket

1 Background

In February, 2011 the Internet Assigned Numbers Authority (IANA) announced that the remaining free blocks of IPv4 addresses had been allocated to the five Regional Internet Registries (RIRs). While the RIRs still have IPv4 addresses to distribute for some period of time, it should be clear that the need for IPv6 is imminent. As further evidence of IPv6 adoption, on June 6, 2012, the Internet Society carried out a World IPv6 Launch Day where participants permanently enabled IPv6 for their products and services. Participating companies included Google, Cisco, Microsoft, AT&T, and many others.

While IPv6 momentum is growing in the Internet and IT space, its adoption in in IP-based Industrial Automation and Control System (IP-based IACS) is still not clear. Since devices in IACS applications typically use private IPv4 addresses not routable on the Internet, global IPv4 address depletion need not be an issue for today's automation architectures. It is more likely that other factors will cause short term IPv6 adoption: government or industry requirements, new applications such as Smart Grid, connectivity via 4G cellular networks, or simply to take advantage of new capabilities inherent in IPv6. We must also plan for a potential explosion of the number of devices installed in industrial applications; the 'Internet of Things' movement calls for ever smaller and simpler IP enabled devices [10] and we should plan for a future where every micro-processor based device has an IP (if not Ethernet) connection. Regardless of which factors ultimately drive IPv6 adoption in the IACS space, it is prudent for EtherNet/IP to be made "IPv6-ready".

In this paper, we discuss the IPv6 adaptation of EtherNet/IP. Before making the adaptation, it is first necessary to understand the typical high-level topology of an EtherNet/IP system, which IPv6 transition method is suitable for EtherNet/IP IPv6 adaptation, which parts of EtherNet/IP specification are related to IP protocol, what impacts on these parts are caused by the IPv6 adaptation and so forth. For these purposes, in section two, the paper first discusses the IPv6 transition method for a typical EtherNet/IP system scenario. In section three, the architectural implications are presented during the IPv6 adoption in EtherNet/IP system. In section four, proposals are advised for the EtherNet/IP specification enhancement for IPv6 adaptation. Then in the following two sections, prototyping experiences are concluded together with an inter-operation demo between multiple vendors. Finally, the paper contents and future work are summarized.

2 EtherNet/IP IPv6 Transition Strategy in Automation System

This section first introduces the IPv6 transition path in the IT field and then takes a typical control system as an example to discuss the specific IPv6 transition strategy in the automation system.

IT Network Transition Path [1]

Considering that a large scale of applications have already been deployed on the IPv4 network and there is no possibility to retire them in short term, the IPv6 application should be incrementally deployed alongside the existing IPV4 infrastructure, including network components and user applications. In practice, the IPv6 migration path could be divided by sequence into three intermediate stages as shown in Figure 1: new appearing IPv6 Islands over legacy IPv4 backbone networks, coexistence of IPv4 and IPv6 backbone networks, legacy IPv4 Islands over IPv6 backbone networks. Obviously the final stage, which is not shown in Figure 1, is that all clients, servers and infrastructure routers are IPv6 only devices.

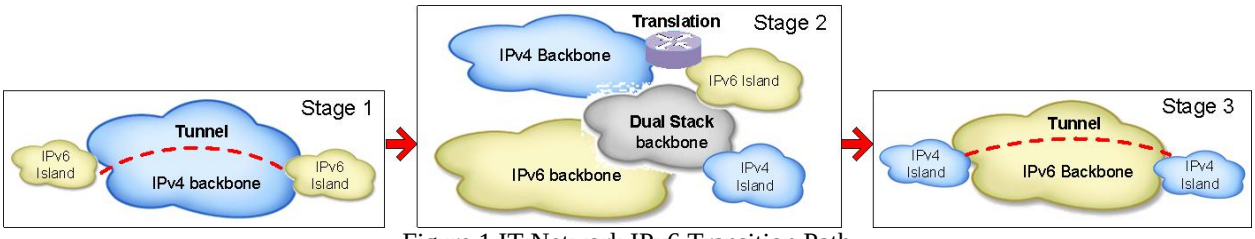

Figure 1 IT Network IPv6 Transition Path

For transition from IPv4 to IPv6, there are three usually used methods available:

- **Dual Stack** – Running both IPv4 and IPv6 on the same devices. Routers and hosts are programmed with both IPv4 and IPv6 implementations to allow them to communicate with both IPv4-only and IPv6-only devices.
- **Tunneling** – Transporting IPv6 traffic through an IPv4 network transparently. IPv6 devices that don't have an IPv6-capable routable path between them are able to communicate by encapsulating IPv6 datagrams within IPv4. The encapsulated IPv4 datagrams would travel across the conventional IPv4 routers.
- **Translation** – Converting IPv6 traffic to IPv4 traffic for transport and vice versa. The devices are designed to accept requests from IPv6 hosts, convert them to IPv4 datagrams, and send the datagrams to the IPv4 destination. The return datagrams are processed similarly.

The "Tunneling" transition method is designed to connect IPv6-only islands of small scale over the IPv4-only backbone networks of large scale as shown in the Stage-1 of Figure 1. Similarly, the "Tunneling" could also be applied to connect IPv4-only islands of small scale over the IPv6-only backbone networks of large scale as shown in the Stage-3 of Figure 1. Different from "Tunneling", the "Translation" method is designed to connect IPv6-only and IPv4-only hosts. As an example shown in the stage 2 of Figure 1, an IPv6 client in an IPv6-only island could access IPv4 servers in the existing IPv4-only backbone network by the translation gateway. When applying these two methods, the "Dual Stack" is actually indispensable and deployed on Edge Components (e.g. Border Routers) between IPv4 and IPv6 networks. In addition to the deployment in border routers, the "Dual Stack" may also be used in networks and hosts for the compatibility with legacy IPv4-only and new IPv6-only devices. As shown in Stage-2 of Figure 1, the devices of "Dual Stack" backbone network could be accessed by either IPv4-only or IPv6-only islands. And more, the intermediate "Dual Stack" networks could also connect the separated networks of the same IP version.

Automation Network Transition Strategy- Dual-Stack

The primary focus in the IT space is to enable many diverse clients to communicate with a relatively small number of servers and this has driven architectural design where the focus has been to ensure that a server can talk to any client. Different from the IT system IPv6 transition path, the automation system has its own special considerations which can be summed up as a relatively small number of clients (controllers/scanners) talking to a large number of servers (I/O devices/adapters) where the focus must be to ensure that a client can talk to any server, leading us towards a dual stack approach in the client.

Figure 2 shows a typical EtherNet/IP automation system over IPv4 network. In this system, the general types of automation devices such as controller, IO module, drive, HMI, and industrial software tool are shown. These devices are divided into two physical subnets connected through an infrastructure network (routers). The subnet of this system can represent a small automation system such as an OEM machine where no router is required and all automation devices are connected together only by an industrial switch. The whole system can represent a larger scale automation system such as a manufacturing work line or even a plant where multiple subnets are connected through an infrastructure network (routers).

Figure 3 shows a transitional EtherNet/IP automation system where IPv4 and IPv6 coexist and the existing IPv4-only devices and the new dual-stack devices coexist. In order to make the new device ready for future IPv6 applications while also compatible with the legacy IPv4-only applications, only the **dual-stack** strategy is adopted for any new IPv6-capable automation device and no IPv6-only device will be introduced in a short term. This approach protects the current manufacturing investments for the end user during the IPv6 transition process.

In this example, the Controller and the Software Tool in subnet 1 and the Drive in subnet 2 are evolved to the hybrid dual-stack devices and all other devices are still IPv4-only devices. All IPv4 CIP communications in the IPv4-only system as shown in Figure 2 are still available in the transitional system. Specifically, the new dual-stack controller in subnet 1 can communicate with the IPv4-only controller in subnet 2 through an IPv4 class 1 IO connection, the new dual-stack software tool can configure the new dual-stack controller through an IPv4 class 3 explicit connection, and the IPv4-only controller in subnet 2 can still communicate with the dual-stack drive through an IPv4 class 1 IO connection too. Meanwhile, the new IPv6 CIP communications are introduced. As shown in Figure3, the hybrid software tool configures the hybrid controller through an IPv6 class 3 connection within the subnet 1, and the new dual-stack controller in subnet 1 controls the new dual-stack drive in subnet 2 through an IPv6 class 1 connection across the two subnets.

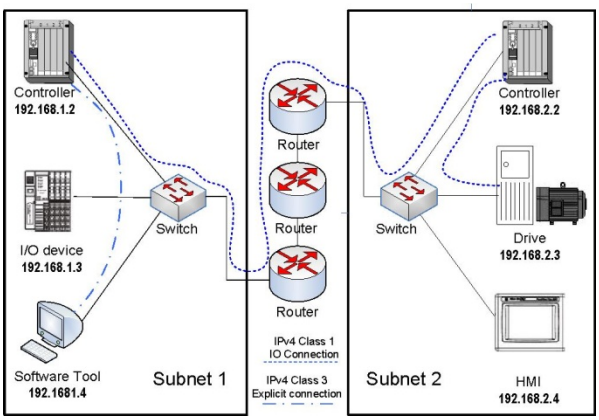

Figure 2 IPv4 EtherNet/IP Automation System

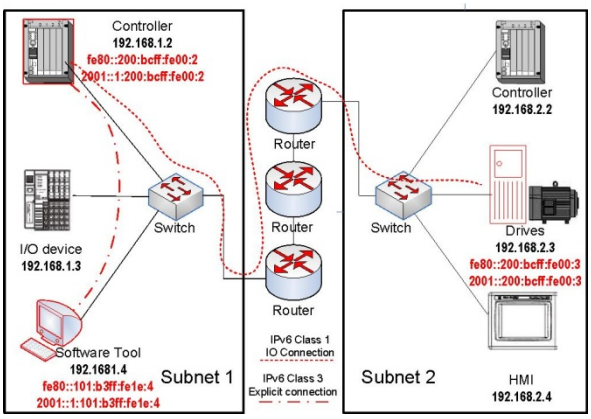

Figure 3 Hybrid EtherNetIP Automation System

For a small automation system like an OEM machine where all automation devices are connected through the layer 2 industrial switch, any two dual-stack hybrid EtherNet/IP devices can establish IPv6 CIP connections without constraints from the network infrastructure at any time. However, for a large automation system where the dual-stack hybrid EtherNet/IP devices are in different subnets, whether the IPv6 CIP communication can be done or not between these dual-stack hybrid devices depends on the network infrastructure (router) capability. In fact, the IT approaches discussed above apply to the cross-subnet CIP communication in the automation system too. Specifically, if the intermediate network is IPv4-only, an IPv6 tunnel is required. If the intermediate network is dual-stack, the two dual-stack hybrid EtherNet/IP devices can communicate via IPv6 protocol natively.

As the final goal of EtherNet/IP IPv6 transition strategy, the system will contain only IPv6-capable devices. However, it should also be noted that the existing legacy IPv4 devices will still remain there for a long time and the transition is a long term evolution. For the transition system, the dual-stack hybrid application devices will play a critical transitional role to enable the coexistence of the legacy IPv4 only devices and the new IPv6 capable devices.

3 Architectural Implications for EtherNet/IP IPv6 Transition System

In this section, we discuss the architectural implications on the automation system and devices from the application perspective of product user and the technical perspective of product developer. We first describe the IPv6 addressing architecture to help understand the IP address changes, and then explain how to set IPv6 address for the automation devices in a system and then describe how to browse the device and how to change IP address focusing on IPv6 impacts. Besides these, we highlight the architecture implications on the product firmware and hardware design from dual-stack support and hybrid application development point of view.

IPv6 Addressing Architecture [3]

Different from IPv4 address format, the IPv6 address is 128-bits in length, and is represented in numeric Hex separated by semicolons in every two bytes, for example, 2001:0:0:0:101:b3ff:fe1e:2222. So when setting an IP address in the IPv6 system, users are required to enter a longer string of address than in the IPv4 system.

An IPv6 unicast address normally consists of 64-bits Network Prefix and 64-bits Interface ID as shown in Figure 4. The Network Prefix normally assigned by routers indicates which subnet the host belongs to and the Interface ID normally derived from MAC address indicates the unique identity of the host network interface on the local link. The Network Prefix of IPv6 protocol substitutes the subnet mask of IPv4 protocol. The users just need to enter a network prefix length instead of a subnet mask value for denoting which network an IPv6 device belongs to. The interface ID derived from MAC address ensures that IPv6 address can identify the device uniquely.

Since the IPv6 address has enough space to encode every particle of sand on the earth, there's no need to concern about the drainage of the address pool. With this prerequisite, every node in the network is allowed to have more than one IPv6 unicast address. Generally, an IPv6 node should have one link-local unicast address and one global unicast address. Of course, some specific applications may require an IPv6 node to have more than one global

unicast addresses for security or other purposes. Although the feature of multiple unicast address on single IPv6 node could potentially improve the security management in the future IPv6 EtherNet/IP system, it may also cause the address selection confusions whereas the fixed addresses are always essential in the automation system.

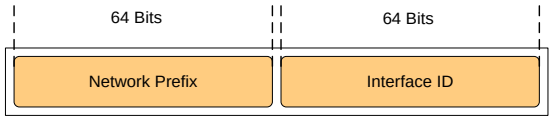

Figure 4 IPv6 Unicast Address Structure

An IPv6 multicast address is composed of a specific prefix containing a new field of the scope definition which is different from that in IPv4 and a group ID representing a group of multicast listeners. Some special multicast addresses are designed for supporting the new features of IPv6. For example, the IPv6 multicast address of FF02::1 deprecates the IPv4 Broadcast address of 255.255.255.255, which means the destinations are all nodes on the local link. Another example, there's a special node-solicited multicast address which is used by Neighbor Discovery protocol to do address resolution, just like the ARP (Address Resolution Protocol) functionality in IPv4.

IPv6 Address Assignment [4] [5]

The IPv6 address assignment mainly points to the unicast address setting on an IPv6 node while the multicast address is automatically allocated by a determined algorithm when creating the multicast IO connection between nodes. Similar to IPv4 address setting, there are two kinds of methods for IPv6 unicast address assignment: manual setting and automatic setting.

There are a number of different ways to set IPv4 address manually varying from one product to another: rotary or DIP switch, special software on PC, web browser, HIM, DHCP/BOOTP on PC etc. Most of these methods still apply to IPv6 unicast address setting. However because IPv6 address is much longer than IPv4 address (128bit vs 32-bit) and its representation format is not intuitive, it is more difficult for the user to input, to label and to remember an IPv6 address. Therefore, IPv6 address manual setting is not desirable and IPv6 address auto-setting becomes a kind of mandatory requirement for the end user.

The very attractive new feature of IPv6 protocol: stateless address auto-configuration based on the Neighbor Discovery protocol provides a very good opportunity to reduce the complexity of IPv6 address setting. Each IPv6 node will be assigned a link-local IPv6 address automatically after power-up. All the IPv6 nodes within a subnet can automatically self-form a network and communicate each other through the Neighbor Discovery (ND) protocol without any configuration. The network prefix of the link-local unicast address is fixed to FE80::/64 and the Interface ID of the link-local unicast address is constructed in modified EUI-64 format[9] and derived from the MAC address of the network interface card, so people can determine the node's link-local IP address easily. At the same time, each IPv6 node can also be automatically assigned a global unicast address only if there is a router on the local link of the node. In this situation, the router will send Router Advertisement of ND protocol to offer the network prefix to the local nodes. Then the node constructs the IPv6 global unicast address with this network prefix and the interface id same to the link-local unicast address. IPv6 defines this as a native feature of IPv6 router and host. In addition to stateless address auto-configuration, DHCPv6 can still be used to automatically configure IPv6 global unicast address of IPv6 node. This requires the node deploys a DHCPv6 client and the network deploys a DHCPv6 server.

Although the automatic IP address assignment relieves users from entering a long string of IPv6 address, it brings another issue about how to identify each device. As a potential solution, a host or domain name of device could be used for identification instead of IP address. Whenever a device automatically gets an IP address, it should register the Name-to-Address mapping in the name resolution server such as DNS. Then in operation, only the name identity appears in user interface and could be resolved to IP address by the name server. This name based approach can also ease migration between IPv4 and IPv6 providing a consistent user experience irrespective of which method is used.

EtherNet/IP Device Browsing

When migrating to IPv6, browsing a hybrid EtherNet/IP network will be significantly different from browsing a pure IPv4 network.

Firstly scanning an IPv6 subnet using the unicast ListIdentity command is impractical because the address space for one IPv6 subnet is huge (normally the Interface ID field of an IPv6 address has 64-bit). In addition, IPv4 remote subnet browsing using the broadcast ListIdentity command (For example, to the destination of 192.168.255.255) cannot be supported because no corresponding mechanism is available in IPv6.

In an IPv6 transitional automation system, both the existing IPv4 devices and the new dual-stack hybrid devices will coexist in one network. Therefore, when browsing this network, the network management software tool must send the query request twice separately for both IPv4 and IPv6 devices. As a result, it must display all IPv4 devices and all IPv6 devices in a single view or in separated IPv4 and IPv6 views. The hybrid dual-stack device will appear twice in the separated display view identified separately by IPv4 and IPv6 address.

When responding to an IPv6 query request, if multiple IPv6 unicast addresses are available, the hybrid device must decide a preferred-to-use IPv6 unicast address and report it in the response. Reporting all available IPv6 unicast addresses may be required too.

Besides the network browsing function, the network management software tool may have the function to change the IP addresses of browsed devices. This requires a new IPv6 graphic interface for the user to input IPv6 addresses and other network configuration information.

Dual-Stack Support

The "Dual Stack" is proposed as IPv6 migration strategy for the automation system and device as discussed in Section 2. Here the dual-stack supportive architecture requirements are discussed to provide a reference to the new IPv6-ready EtherNet/IP product development.

Figure 5 shows two different firmware architectures for IPv6 EtherNet/IP products. The left architecture employs the totally separated, independent TCP/IP stack, EtherNet/IP stack and EtherNet/IP application firmware for both IPv4 and IPv6. This architecture may be suitable for adding the IPv6 functionality to the existing IPv4-only device with nothing change to the existing IPv4 firmware, but it is not recommended because it needs more memory, more CPU processing power, and more code maintenance work. The right architecture is our recommended one for all new EtherNet/IP device development. It employs the real dual-stack we refer to. The real dual-stack architecture has the single TCP/UDP layer over the IPv4 and IPv6 protocol and provides the universal socket interfaces to the upper application layer. As an example, Windows XP uses the left architecture while Windows 7 uses the right architecture [11].

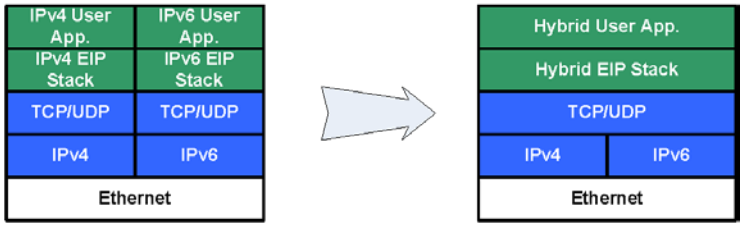

Figure 5 Software Architecture for Hybrid EtherNet/IP Product

When doing a new EtherNet/IP embedded device development, if you want your firmware and hardware platform ready for future hybrid application upgrade, keep in mind to select TCP/IP stack with real dual-stack architecture and to select the hardware platform with the enough memory and MCU processing power for future hybrid EtherNet/IP application.

When doing a hybrid software development, you need to pay attention to the IPv6 capability of OS. Taking Windows operation system as an example, WinXP does not support the IPv6 well but Win7 has better IPv6 support.

Hybrid Application Support

Based on the "Dual Stack", the separated IPv4 and IPv6 versions of applications can be combined together to form the "Hybrid Application". The concept is shown in Figure 5. The "Hybrid Application" includes the top two layers of the hybrid EtherNet/IP communication stack and the hybrid product specific application that are of concerns to the automation product developers and users.

At the hybrid product specific application layer, the protocol independent functions should be constructed for processing the application specific tasks related to both IPv4 and IPv6 protocol. And also the protocol independent data structure should be used to store IP address related data. For achieving this purpose, the universal socket address structure and functions of dual stack could be utilized (Refer to Section 5 for more details). Especially for the hybrid software application, a new graphic user interface is required to allow the user to input both IPv4 and IPv6 address.

As for the EtherNet/IP communication stack layer, it interfaces to the downward TCP/IP stack layer via universal sockets of the dual stack, and it also interfaces to the upward product specific application layer via EtherNet/IP stack APIs. In order to make the hybrid EtherNet/IP stack easy to use, similar to the universal sockets of dual stack, the EtherNet/IP stack APIs should be independent of IP protocol versions too. Besides the external interfaces to the up and down layers, the EtherNet/IP communication stack itself must be enhanced to support both IPv4 and IPv6 communication according to the enhanced EtherNet/IP specification for IPv6. Refer to the next section for the discussion on the EtherNet/IP specification enhancement.

4 EtherNet/IP Specification Enhancement[7]

The current edition of the EtherNet/IP specification (1.13 at the time of this paper) supports IPv4 only. The protocol embeds IPv4 addresses in a number protocol messages, uses IPv4 multicast for I/O data transfer, and includes IPv4 addresses and other IPv4 specifics in application objects. These and several other protocol aspects must be modified or extended in order to support EtherNet/IP communications over IPv6.

The following sections identify areas in the EtherNet/IP specification impacted by IPv6, along with a number of solution proposals. Ultimately the exact specification changes must be developed and confirmed by the EtherNet/IP System Architecture SIG within ODVA.

List Identity in Encapsulation Protocol for Network Browser Function
EtherNet/IP deploys the ListIdentity service in the encapsulation protocol for EtherNet/IP network browsing function. In IPv4 the software tool sends the ListIdentity request to devices using either unicast or broadcast and the device reports its IP address (only one) and device identity information in the response using unicast. When migrating to IPv6, the ListIdentiy service must deal with both IPv4 and IPv6 protocol and needs the following modifications:

- Utilize the IPv6 link-local all-nodes multicast instead of the IPv4 broadcast to process the ListIdentity request on local network.
- Modify the ListIdentity response format definition, or define a new response item for IPv6, to remove the IPv4 address specifics.
- Define behavior for responding when the device has multiple IPv6 addresses.

Forward Open Request/Response for Creating Multicast I/O Connection
When creating multicast I/O connections, the forward open request/response must carry a Sockaddr Info Item to exchange the multicast address information used for establishing the connection. The current Sockaddr Info item defined in Chapter 2 of the EtherNet/IP specification does not have enough size to contain IPv6 address. The proposed solution is to retain the existing item type codes, and define a new family type AF_NET6 to support IPv6 address format. The Sockaddr Info Item can then support both IPv4 and IPv6 multicast addresses. Multicast I/O connections between IPv4 devices would use the AF_INET family, and connections between IPv6 devices would use the AF_INET6 family.

TCP/IP Interface Object for Network Configuration
TCP/IP interface object abstracts all configurations and behaviors which are provided by TCP/IP network and used in the EtherNet/IP protocol. Because of the significant architectural changes of IPv6, this object will need significant changes too. Three general approaches are studied for IPv6 migration regarding to this object.

- Improve the existing object to handle both IPv4 and IPv6 configuration.
- Create a new object to handle both IPv4 and IPv6 configuration and deprecate the existing object.

- Create a new dedicated object to handle IPv6 configuration and keep the existing one to handle IPv4 configuration.

The second option has the advantage of providing an opportunity to address a number of issues with the current object. The third option matches the dual-stack strategy very well. It is recommended that the SIG consider and choose between options two and three.

The contents of IPv4 TCP/IP interface object contain the configurations for IP network (e.g. IP address, sub network, and gateway etc.), multicast, ACD, safety, and quick connect. Most of these configurations will be impacted when migrating to IPv6. The new IPv6 TCP/IP interface object must take care of the following changes:

- Network configuration: Multiple IPv6 unicast addresses (e.g. one link-local address and at least one global address), and IPv6 unicast address selection. IPv6 network prefix length instead of IPv4 network mask. New 16-byte data format to store all IPv6 unicast addresses.
- Multicast configuration: Term change from TTL (Time to Live) to HL (Hop Limit). New logic-based multicast scope definition. New 16-byte data format to store the multicast start address for IPv6 multicast address manual allocation.
- DAD (Duplicate Address Detection) diagnostic information: new ND protocol (Neighbor Solicited Message and Neighbor Advertisement Message) is used for IPv6 DAD. Therefore, the IPv4 ARP message for the address conflict diagnostic in this object needs to be updated.

UNID of EtherNet/IP Safety [8]

The device Unique Identifier in CIP safety specification consists of a 4-byte node ID and 6-byte Safety Network Number (SNN). For EtherNet/IP safety, the node ID is IPv4 address of the devices and the SNN is assigned by the safety network configuration tool. The SNN is either auto-generated by the software tool or manually set by people. Both IPv4 address and SNN are stored in TCP/IP interface object. Refer to [8] for more details on UNID definition. When migrating to IPv6, because 16 bytes of IPv6 address is much larger than 4 bytes of node ID in the safety UNID, the IPv6 address cannot be used directly to construct the UNID. A new mechanism must be developed that ensures system-wide UNIDs across both IPv6 and IPv4 devices. .

IP Multicast Usage in EtherNet/IP for Produce/Consume Model

At the time of writing this paper, multicast work is still under progress. However, because multicast is a key function of EtherNet/IP the impacts are still highlighted even if no change proposals are made.

Each device needs to allocate a multicast IP address for producing data over a multicast CIP connection. IPv4 EtherNet/IP protocol defines a special algorithm to do this. Because of IP address space and architecture changes, the IPv4 multicast address allocation method might not work for IPv6, so the new method may be required. One potential improvement is to break up the limitation of maximum 32 multicast addresses available for one device because the address space is not an issue for IPv6.

IPv4 multicast uses IGMP protocol while IPv6 multicast uses MLD protocol. This protocol change requires the clarification of EtherNet/IP device behavior related to using MLD protocol. This impacts the host, switch and router. For example, a device may need to use different socket interfaces to perform IPv6 multicast communication, and the linked switch may need to support MLD snooping function to reduce the multicast traffic.

IPv4 just simply uses Time to Live (TTL) in IP header to control the multicast traffic propagation on network. When migrating to IPv6, this mechanism is still there but TTL is changed to a better term: Hop Limit (HL). Moreover, IPv6 introduces the scope definition within the multicast address architecture. This will improve the multicast traffic control not just based on the physical topology (how many routers the traffic can cross) but also based on the logic participation of network (which site the traffic can go to). IPv6 EtherNet/IP may need to consider how to incorporate this new feature.

IP Address Conflict Detection / Duplicate Address Detection

At present the EtherNet/IP specification defines usage of IPv4 Address Conflict Detection (ACD), as defined in RFC 5227, which is specific to IPv4. IPv6 in contrast defines Duplicate Address Detection (DAD) as part of Neighbor Discovery. Since IPv4 ACD clearly applies to IPv4 only, no changes or modifications are required. IPv4 ACD is used for IPv4 addresses. For IPv6 further investigation is needed to determine whether any EtherNet/IP specific behaviors are needed to be linked with IPv6 DAD behavior (e.g., LED behavior, object attributes, etc.)

Quality of Service
Quality of Service (QoS) as defined in the EtherNet/IP specification makes use of standard mechanisms (DiffServ and IEEE 802.1Q) that are independent of IPv4 and IPv6. Only minor changes are needed to account for the different naming of the IPv4 and IPv6 header fields that carry the DSCP values.

Device Level Ring (DLR) and Beacon Redundancy Protocol (BRP)
A number of DLR and BRP protocol messages have embedded IPv4 addresses (e.g., for Supervisor IP address). It should be possible to modify the message format to allow for either IPv4 or IPv6 address in a way that is backwards compatible with existing IPv4 implementations.

In addition, a number of DLR and BRP Object attributes include IPv4 addresses. Supporting IPv6 addresses will most likely require new versions of the objects with attributes that can hold either IPv4 or IPv6 addresses.

EtherNet/IP Quick Connect
Quick Connect includes references to IPv4 ACD. At a minimum the address conflict detection references need to be made applicable to either IPv4 or IPv6. Further research is required to determine whether there is additional impact on Quick Connect behavior related to IPv6 address assignment and neighbor discovery (including IPv6 DAD).

General Host Requirements for IPv6
Aside from changes and extensions to existing EtherNet/IP mechanisms, further research is needed regarding IPv6 features and behaviors and whether or how they should apply to EtherNet/IP devices. This includes identifying specific RFCs and how those RFCs would apply to EtherNet/IP. As an example, RFC 4294 (IPv6 Node Requirements) specifies that IP Security Architecture (IPsec) MUST be supported. It is unclear whether such a requirement is appropriate for EtherNet/IP devices. Similar issues may exist with other RFCs.

5 EtherNet/IP IPv6 Development Practice

A hybrid EtherNet/IP application software is developed under Win7 OS with Visual Studio 2008 based on Pyramid Solution's EtherNet/IP stack. This application software combines the function of network browsing, UCMM, Class 3, and Class 1 unicast and can simulate the role of network browser software, EtherNet/IP scanner, and EtherNet/IP adapter. The concept of IPv6 adaptation of EtherNet/IP was proved and the know-how on IPv6 development was built through this application software development. Moreover, limiting porting of this to the embedded hardware platform of the existing products using other RTOS, TCP/IP stack and EtherNet/IP stack has proved the portability of this approach. This section shares the important development practice we have learnt behind the high level architecture implications. Note that we focus on IPv6 impacts, not on EtherNet/IP protocol itself.

Dual Stack and Universal Socket Interfaces
A TCP/IP stack with real dual-stack architecture should be selected to support the hybrid EtherNet/IP application development (Refer to Section 3 for more details). The dual-stack TCP/IP stack provides universal socket interfaces including the protocol independent socket address data structure and socket functions.

There are four types of socket address structures as shown in Figure 6 which might be used in the hybrid application development. The data structure sockaddr_in is for IPv4 only application, the data structure sockaddr_in6 is for IPv6 only application, the data structure sockaddr_storage is a general structure for both IPv4 and IPv6 application, and the data structure sockaddr is often used as a pointer parameter of socket functions to cast the pointers of the above different socket address structures for both IPv4 and IPv6 application into the same pointer type. Note that the first two bytes of all these four data structures are address family information that is always accessible in any condition. The practice of using these four data structures for the hybrid EtherNet/IP application is that using sockaddr_storage to store the socket address information, casting sockaddr_storage to sockaddr_in or sockaddr_in6 according to the address family field for the internal IP protocol specific processing, and casting the pointer of sockaddr_storage to the pointer of sockaddr when calling socket functions to communicate.

The socket functions for TCP/UDP socket creating,TCP/UDP receiving and sending control are the same to that in IPv4. The IP address related functions such as address conversion functions, and address resolution functions are changed. Specifically, two new universal address conversion functions of inet_pton() and inet_ntop() are added to replace the old functions of net_addr(), inet_aton(), and inet_ntoa(). And also two address resolution functions of getaddrinfo() and getnameinfo() are added to replace the old functions of gethostbyname() and gethostbyaddr(). It is recommended to use the new universal functions for the hybrid EtherNet/IP applications.

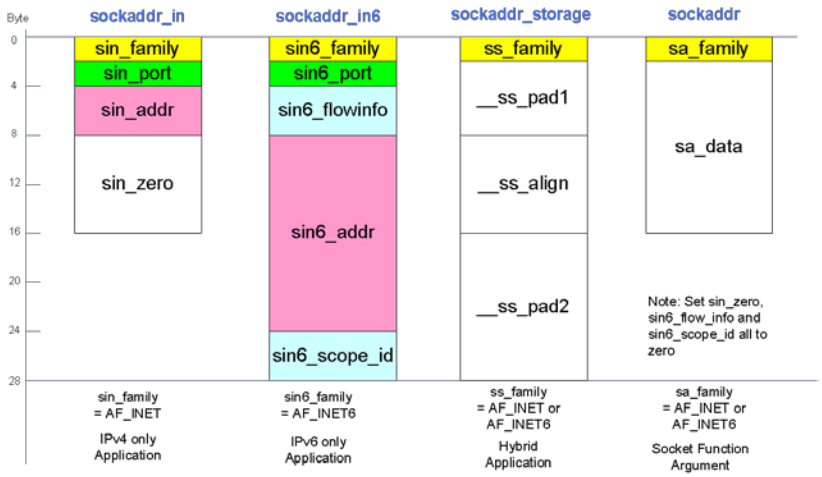

Figure 6 Socket Address Data Structure Illustration

IP Address, Socket Structure and Function Changes in Application and EtherNet/IP Stack Code

Beside the enhancement on the EtherNet/IP protocol as discussed in Section 4, the major changes to migrate the IPv4 only code to the hybrid IPv4-IPv6 code are about IP address, socket structure and function call. The following table provides a summary on these aspects.

Table 1 IPv6 Migration Efforts Analysis from Code Level

	IPv4 only code	Change to IPv4-IPv6 Code	Impact Analysis
Global Variable	All UINT32 of IP address, in_addr and sockaddr_in variables.	Change variable data type from UINT32, in_addr and sockaddr_in to sockaddr_storage.	Impact where using this variable (**big**)
Function interface	UINT32 IP address, in_addr and sockaddr_in as argument of function. UINT32 IP address, in_addr and sockaddr_in as return value of function.	Redefine function interface using sockaddr_storage as argument or return value. New implementation of function body including protocol decision and IPv6 related logic.	Impact the function itself and where using this function (**big**)
Function body	UINT32 IP address, in_addr and sockaddr_in as local variable. Function call of net_addr(), inet_aton(), inet_ntoa() or gethostbyname().	Change local variable data type from UINT32, in_addr and sockaddr_in to sockaddr_storage. Use universal address conversion function inet_pton() and inet_ntop(), and address resolution function getaddrinfo().	Only impact the function itself (small)

IP Protocol Selection

A hybrid EtherNet/IP application must make a decision on which protocol it should use to communicate with other nodes. In short, the hybrid EtherNet/IP application chooses the protocol according to its IP address setting and the IP address format of the peer device it wants to communicate with as shown in Figure 7. Specifically, if only IPv4 address is configured, then IPv4 protocol is used. If only IPv6 address is configured, then IPv6 protocol is used. If both IPv4 and IPv6 addresses are configured, then if it is an originator, it depends on the target device's IP address configuration available in originator (if both IPv4 and IPv6 addresses are available for the target device, either IPv4 or IPv6 address can be used and more investigation on the user preferred default behavior is required); if it is a target, it depends on the coming protocol from the originator. Assume an example that a hybrid ControlLogix controller (originator) communicates to a hybrid IO device (Target). Before the CIP connection is created to the IO device, the IP address of that IO device must be first configured in the controller through RSLogix5000. The end user decides which protocol is to be used and then configures the IO device in the RSLogix5000 via IPv4 or IPv6. After downloaded with the RSLogix5000 project, the controller will choose the protocol according to this pre-configured

IP address format to communicate with that IO device. The IO device will choose the protocol according to address family information in the coming socket for detecting a connection creating request from the controller. If a host name of this hybrid IO device is configured in the ControLogix controller, the end user cannot control the IP address usage since IP addresses are retrieved from DNS server in the background. If both IPv4 and IPv6 addressed are returned from DNS server, the ControLogix controller must adopt a default behavior (for example, IPv6 preference) internally to select an IP address for the hybrid IO device.

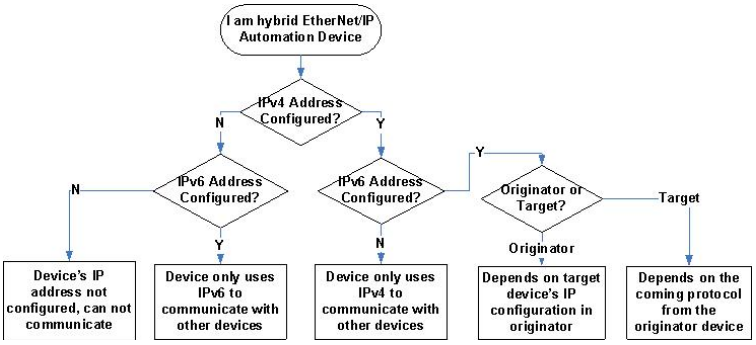

Figure 7 IP Protocol Selection Flow Chart for Hybrid EtherNet/IP Application

IPv6 Source Address Selection

IPv6 node may have multiple IPv6 unicast IP addresses available, for example, one link-local and one global address. The hybrid EtherNet/IP application must make a decision on which IPv6 address is used as source address when it sends the packet to the network. TCP/IP stack has its default source address selection algorithm (refer to RFC3484 [6]). One decision rule according to RFC3484 is that the device prefers the source address with the appropriate scope of the destination address. In the case of EtherNet/IP network browsing, when sending an IPv6 multicast ListIdentity request, the network browsing software will use the default link-local unicast address if the source address is not specified. This is because the destination address is link-local multicast address. As a result, the ListIdentity request receiver whose source address is not specified will use a link-local IPv6 unicast address as source address for responding even if it has a global IPv6 unicast address. This IPv6 source address selection method is not suitable for industrial network browsing application.

So EtherNet/IP specification must adopt an IPv6 source address selection algorithm in the application layer to override the default IPv6 source address selection behavior in the TCP/IP layer. Figure 8 proposes one IPv6 source address selection method which has been verified in our hybrid EtherNet/IP application software. The device always has a link-local unicast IP address. If no global IPv6 unicast address is available, then the link-local unicast address is used as source address for all CIP communication. If the global IPv6 unicast address (only one) is available, then this global IPv6 unicast address will be used as source address for all unicast CIP communication (UCMM, Class 3, Class 1 Unicast, ListIdentity Response and ListIdentity Request in unicast etc.) and all class 1 multicast communication while the link-local unicast address is only used as source address for sending the link-local all-node multicast ListIdentity request. Although Figure 8 also shows the case of multiple global unicast IPv6 addresses available, we propose that only one global unicast IP address shall be allowed for the automation devices to reduce the complexity of the address configuration and selection.

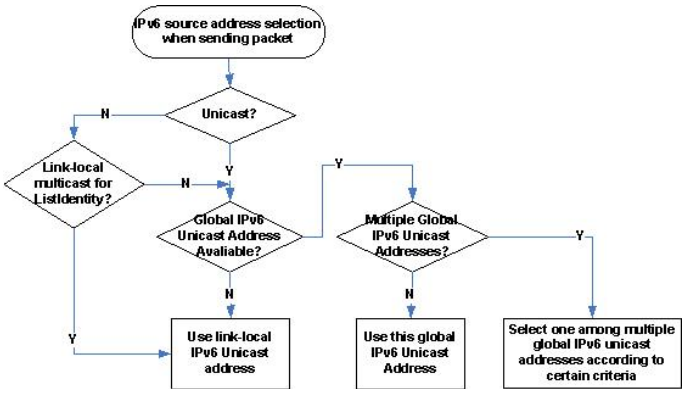

Figure 8 IPv6 Source Address Selection Flow Chart

Sockets Created for EtherNet/IP Communication

This part targets the EtherNet/IP stack developer. It provides a summary on how the EtherNet/IP stack interfaces to the TCP/IP universal sockets and what sockets are created for what purpose and when to create these sockets.

The EtherNet/IP UCMM or Class 3 server need to create a passive listening socket of IPv4 and IPv6 version to detect the open session request from the client for creating a session for the further CIP communication. The EtherNet/IP UCMM or Class 3 client and server will create the TCP sockets of IPv4 and IPv6 versions for UCMM or Class 3 communication during run-time operation. The EtherNet/IP Class 1 originator and target will create the UDP sockets of IPv4 and IPv6 versions for class 1 communication during run-time operation.

Besides, the IPv4 device must create a broadcast socket and the IPv6 device must create a link-local all-node multicast socket for processing the ListIdentity service of the encapsulation protocol for the network browsing function.

Table 2 Created Sockets Summery for Hybrid EtherNet/IP Stack

IP Version	EtherNet/IP Function	Created Sockets	Client or Originator Socket Function	Server or Target Socket Function
IPv4	Network Browser	IPv4 BroadCast socket (one per interface)	Send broadcast packet and receive unicast packet. Created during start-up.	Receive broadcast packet and send unicast packet,created during start-up
	UCMM or Class 3	IPv4 Listen Socket (one per interface)	N/A	Receive new session request from Client for creating a TCP socket. Created during start-up
		IPv4 TCP socket (multiple per interface)	Send UCMM or Class 3 request and receive UCMM or Class 3 response. Created when application starts UCMM or Class 3 communication	Receive UCMM or Class 3 request and send UCMM or Class 3 response. Created when detecting a RegisterSession Request from Client
	Class 1 Unicast	IPv4 UDP socket (multiple per interface)	Produce the output data to target and consume the input data from target. Created when receiving a successful ForwardOpen response	Produce the input data to originator and consume the output data from originator. Created when receiving a ForwardOpen request
IPv6	Network Browser	IPv6 Link-local Multicast socket (one per interface)	Send link-local multicast packet and receive unicast packet, created during start-up	Receive link-local multicast packet and send unicast packet,created when start-up
	UCMM or Class 3	IPv6 Listen Socket (one per interface)	N/A	Receive new session request from Client for creating a TCP socket, created during start-up
		IPv6 TCP socket (multiple per interface)	Send UCMM or Class 3 request and receive UCMM or Class 3 response, created when application starts UCMM or Class 3 communication	Receive UCMM or Class 3 request and send UCMM or Class 3 response, created when detecting a RegisterSession Request from Client
	Class 1 Unicast	IPv6 UDP socket (multiple per interface)	Produce the output data to target and consume the input data from target. Created when receiving a successful ForwardOpen response	Produce the input data to originator and consume the output data from originator. Created when receiving a ForwardOpen request

6 Proof of Concept EtherNet/IP IPv6 Demonstration

An EtherNet/IP IPv6 demonstration is built to prove the concept of IPv6 adaptation of EtherNet/IP. The demo intends to involve vendors of Rockwell automation, HMS and those potential participants showing interests to test

the inter-operation and to gain more industrial interests. The demo can also unveil the potential issues and validate the proposed approaches for IPv6 adaptation of EtherNet/IP before putting them into the EtherNet/IP specification.

At current stage, the inter-operation demo has the architecture shown as in Figure 6. The ControlLogix from Rockwell Automation and the IE3000 from Cisco are used as the controller and switch. Hybrid EtherNet/IP PC Scanner and PC Adapter are running on PCs. Besides, a workstation PC is running Wireshark and RSLogix5000 software for analyzing Ethernet traffic and programming control programs.

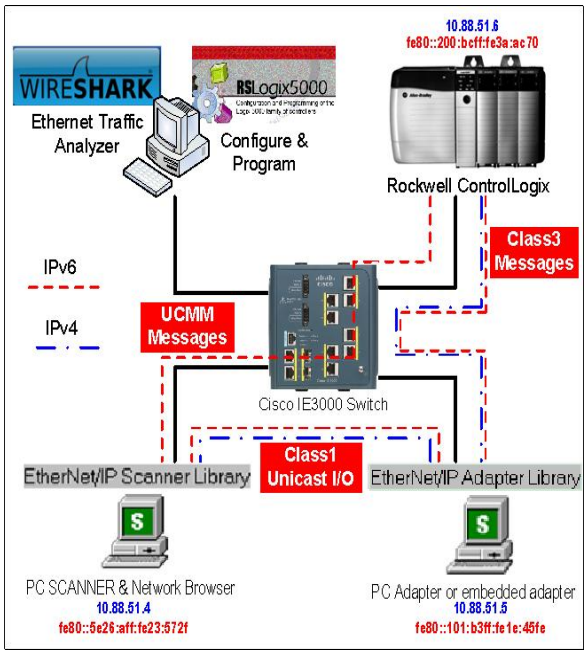

Figure 9 EtherNet/IP IPv6 Demo Architecture

In the demo, the following functions related to IPv6 protocol are tested:

- Network Browsing. The hybrid EtherNet/IP application software works as network browser to browse all EtherNet/IP devices in network and create a display view for the browsed devices. Both the legacy IPv4-only devices and the dual-stack hybrid application devices are displayed in the view. The hybrid devices are displayed twice identified separately by IPv4 and IPv6 address.
- IPv6 Address Setting. This function simulates the device commissioning to set IP address on each automation devices. Users are asked to select from the available IPv4 or IPv6 IP addresses on the host and assign them to the PC scanner or PC adapter during startup.
- IPv6 Source Address Selection. The global IPv6 unicast address preference algorithm is tested with the PC scanner or PC adapter.
- UCMM Client and Server. The IPv6-based UCMM communication is tested between network browser and ControlLogix. This function simulates the processing of unconnected explicit messages from the scanner or software tools to devices to get the devices' information or make configuration.
- Class 3 Client and Server. IPv6-based Class3 connection is tested between ControlLogix and PC adapter. This function simulates the processing of the connected explicit messages from the scanner or software tools to devices to get devices' information or make configuration.
- Class 1 Originator and Target. Multiple IPv6-based Class 1 Unicast I/O connections are tested between PC scanner and PC adapter.
- Hybrid EtherNet/IP application and IPv4 and IPv6 coexistence. Both IPv4 and IPv6 class 3 connections are created between ControlLogix and PC adapter and both IPv4 and IPv6 class 1 connections are created between PC scanner and PC adapter to test the IPv4-IPv6 coexistence of system and the hybrid application of device.

7 Summary

Because EtherNet/IP has been built in strict accordance with the ISO/OSI 7 layer model and exists almost exclusively at the application layer, it is largely abstracted from IP which lives at layer 3. We have demonstrated incorporation of IPv6 into the EtherNet/IP specification will enable smooth migration paths for both device vendors and end users from IPv4 to IPv6 based solutions at a market driven rate. Many existing devices will have practical upgrade paths with only firmware changes.

This paper discussed the IPv6 adaptation of EtherNet/IP automation system. Some preliminary experiences have been obtained with regards to: the IPv6 migration strategy in automation system, implications for the current system architecture, enhancement proposal of EtherNet/IP specification for IPv6 adaptation, implementation issues and approaches by prototyping and demonstration. An inter-operation demo with basic functionality is also presented for the preview of the IPv6 EtherNet/IP system in some future day. In the next step, it is still desired to explore IPv6 adaptations of more advanced functionality of EtherNet/IP system, such as Class1 multicast I/O communication, QoS, DLR, IPsec etc. And also the demo communication should go beyond link-local scope to simulate the real plant scenarios. In this situation, more network infrastructure issues in the IPv6 field such as routing configuration of Routers, Multicast snooping etc. are worth studying. Furthermore, the DHCPv6, DNS for IPv6 or other IP suite protocols might improve the system performance as well.

References

[1] The IPv6 Transition by Sean Wilkins - January 4, 2012, http://www.petri.co.il/ipv6-transition.htm
[2] RFC2460 Internet Protocol, Version 6 (IPv6) Specification
[3] RFC4291 IPv6 Addressing Architecture
[4] RFC4861 Neighbor Discovery for IPv6
[5] RFC4862 IPv6 Stateless Address Autoconfiguration
[6] RFC3484 Default Address Selection for IPv6
[7] THE CIP NETWORKS LIBRARY Volume 2 EtherNet/IP Adaptation of CIP Edition 1.13April 2012
[8] THE CIP NETWORKS LIBRARY Volume 5 CIP Safety Edition 2.5 November 2011
[9] Guidelines for 64-bit Global Identifier (EUI-64™) Registration Authority, IEEE
[10] “Internet of Things” by Michael Chui, Markus Löffler, and Roger Roberts, McKinsey Quarterly March 2010
[11] IPv6 Transition Technologies, Microsoft Corporation, Published: October 2003, Updated: February 2008